NASA CR-134457

N73-30141

NASA-CR-134457) NASIS DATA BASE MANAGEMENT SYSTEM: IBM 360 TSS IMPLEMENTATION. VOLUME 3: DATA SET (Neoterics, Inc., Cleveland, Ohio.) 199 p HC $12.00

CSCL 09B G3/08 Unclas 13479

NASIS DATA BASE MANAGEMENT SYSTEM - IBM 360 TSS IMPLEMENTATION

III - DATA SET SPECIFICATIONS

NEOTERICS, INC.

prepared for

NATIONAL AERONAUTICS AND SPACE ADMINISTRATION

NASA Lewis Research Center
Contract NAS 3-14979

199

TABLE OF CONTENTS

TOPIC B.1 - DATA BASE EXECUTIVE

A. DATA SET NAME:

DBPL/I Diagnostics

B. CREATED BY:

DB Preprocessor Function

C. TYPE OF FILE:

(4) Table

D. ORGANIZATION:

Keyed List

E. KEY IDENTIFIER (CONTROL FIELD):

Each diagnostic comment has a five-character identification key having the form: DBnnn, where nnn is a unique identification number.

F. RECORD LENGTH:

Variable

G. BLOCKING FACTOR:

Not Applicable

H. PURPOSE:

DBPL/I Diagnostic Comments are generated into mainline source programs by the DB preprocessor function (see Section IV, Topic B.1 of the DMB).

I. DBPL/I DIAGNOSTIC COMMENTS:

DB001 INITIALIZATION COMPLETE.

Informative - the % INCLUDE DB statement has been successfully processed.

DB002 'DB' FOUND WITHOUT ARGUMENT. IT IS A RESERVED IDENTIFIER.

Severe error - a DB preprocessor function reference has no parenthesized argument.

DB003 MISSING LEFT PARENTHESIS.

Severe error - a DB preprocessor function reference does not begin with double left parentheses. Processing of this DB reference was abandoned because the closing right parenthesis would not be able to be found.

DB004 ARGUMENT ABANDONED. TOO MANY DBPL/I ERRORS.

Error - more than four errors have been noted from one DB preprocessor function reference so it is being abandoned. This diagnostic may arise when the right parenthesis are missing at the end of the argument (PL/I passes the remainder of the source program to the DB function).

DB005 EXTRANEOUS TEXT IGNORED.

Error - if this message immediately follows DB009, then the statement has been processed properly but additional clause(s) other than comments intervene before the semicolon. Verify that the statement has its own semicolon. If this message follows a diagnostic other than DB009, it means merely that part of the statement was ignored.

DB006 MISSING SEMICOLON OR COLON. 'text'

Severe error - the right parenthesis at the end of a DB preprocessor function reference has been encountered unexpectedly. The label or statement "text" shown was ignored.

DB007 MISSING SEMICOLON. 'text'

Severe error - the right parenthesis at the end of a DB preprocessor function reference has been encountered unexpectedly. The statement "text" shown was ignored.

DB009 DBPL/I STATEMENT: 'text; comments'

Informative - a non-null statement has been found. The "text" shown is the statement as internally rearranged into a standard format without embedded comments for further analysis. The "comments" shown

are those extracted from the statement.

DB011 STATEMENT FOLLOWS FINISH.

Severe error - the statement has been ignored because it follows the DB((FINISH;)) reference.

DB013 STATEMENT HAS 'n' MORE LEFT PARENTHESES THAN RIGHT PARENTHESES.

Severe error - the statement semicolon has been found but the parentheses are unbalanced. The statement was ignored.

DB015 UNKNOWN STATEMENT KEYWORD: 'word'.

Severe error - the 'word' shown is not the first word of a DBPI/I statement. The statement was ignored.

DB017 INVALID LISTERROR ACTION.

Severe error - an ON LISTERROR statement was ignored because its action clause was neither SYSTEM nor GO TO.

DB019 INVALID ERRORFILE.

Severe error - an ON ERRORFILE statement was ignored because its filename was longer than eight characters or was not terminated by a right parenthesis.

DB021 INVALID ON CONDITION.

Severe error - an ON statement was ignored because it was neither ON LISTERROR nor ON ERRORFILE. These are the only ON statements recognized by the DB preprocessor function.

DB023 MISSING LIST POINTER.

A GET LIST or PUT LIST statement was ignored because it did not contain a required parenthesized list pointer.

DB025 INVALID GET LIST CLAUSE.

Severe error - a GET LIST statement was ignored because it did not contain either a KEY(0) or a KEY INTO clause.

DB026 INVALID PUT LIST CLAUSE.

Severe error - a PUT LIST statement was ignored

because it did not contain required INTERNAL KEY FROM clauses.

DB027 INVALID FILE.

Severe error - a statement having a FILE clause was ignored because its filename was longer than eight characters or was not terminated by a right parenthesis.

DB028 MISSING LIST.

Severe error - a SET statement was ignored because it did not contain a required LIST clause.

DB029 MISSING FILE.

Severe error - a statement that should have a FILE clause was ignored because it did not have one.

DB030 MISSING SIZE.

Severe error - a SET LIST statement was ignored because it did not contain a required SIZE clause.

DB031 INVALID ON ACTION.

Severe error - an ON ERRORFILE statement was ignored because its action clause was neither SYSTEM nor GO TO.

DB032 MISSING 'LIST' OR 'KEY' CLAUSE.

Severe error - a GET FILE statement was ignored because it did not contain either a LIST or a KEY clause.

DB033 MISSING FIELD CLAUSE.

Severe error - a statement that should have a FIELD clause was ignored because it did not have one.

DB034 MISSING 'LIKE LIST'.

Severe error - a SET LIST SIZE statement was ignored because it did not contain a required LIKE LIST clause.

DB035 MISSING INTO.

Severe error - a GET FIELD statement was ignored because it did not contain a required INTO clause.

DB037 MISSING FROM.

Severe error - a PUT or REPUT statement was ignored because it did nct contain a required FROM clause.

DB039 MORE ITEMS THAN FIEIDS.

Severe error - excess INTO or FROM items in a GET, PUT or REPUT FIELD statement were ignored.

DB041 MORE FIEIDS THAN ITEMS.

Severe error - excess fieldnames in a GET, PUT or REPUT FIELD statement were ignored because the INTO or FROM clause has too few items.

DB043 INVALIC CPEN CLAUSE.

Severe error - an OPEN statement has been abandoned because one of its substatements has an invalid or out of order FILE, TITLE, access or function clause.

DB045 INVALID READ CPTION(S).

Severe error - a READ statement has been ignored because it has an invalid or out of order file-positioninq or NOLOCK option.

DB047 INVALID CLOSE SYNTAX.

Severe error - a CIOSE statement has been abandoned because one of its substatements has an invalid or cut of order FILE or ERASE clause.

DB049 MISSING FROM.

Severe error - a WRITE statement has been ignored because it did nct contain a required FROM clause.

DB051 MISSING KEYFROM.

Severe error - a LOCATF statement has been ignored because it did not contain a required KEYFROM clause.

DB053 LIST OPTION MISSING.

Severe error - a FREE statement has been ignored because it did nct contain a required LIST option.

DB055 INVALID LIST.

Severe error - a FREE LIST statement has been ignored because it did not contain a parenthesized set of list-pointers.

DB057 'filename' FILE HAS STATEMENT DIAGNOSTIC(S).

Severe error - the FINISH statement has not generated a Mainline File Control Block declaration (see Section III, Topic B.4 of the DWB) for the 'filename' shown because errors in its use have been previously detected. Note that a missing MFCB declaration will yield "undefined qualified name" diagnostics from the PL/I compiler for correct DBPL/I statements using the 'filename'.

DB059 'filename' FILE HAS NON-INPUT USE(S).

Informative - the 'filename' shown has a use that may conflict with the INPUT file function attribute.

DB061 'filename' FILE HAS NON-OUTPUT USE(S).

Informative - the 'filename' shown has a use that may conflict with the OUTPUT file function attribute.

DB063 'filename' FILE HAS NON-UPDATE USE(S).

Informative - the 'filename' shown has a use that may conflict with the UPDATE file function attribute.

DB065 'filename' FILE REQUIRES UPDATE ATTRIBUTE.

Informative - the 'filename' shown has a use that requires the UPDATE file function attribute, but this compilation does not contain a valid OPEN...UPDATE statement for the 'filename'.

DB067 'n' DBPL/I ERRORS.

Informative - the FINISH statement has been processed and 'N' errors were previously detected. The programmer should find and analyze the 'N' DBPL/I diagnostic comments.

TOPIC B.2 - DATA BASE EXECUTIVE

A. DATA SET NAME:

DBPL/I - DBPAC Interface

B. CREATED BY:

DB Preprocessor Function

C. TYPE OF FILE:

(4) Table

D. ORGANIZATION:

Documentary Table

E. KEY IDENTIFIER (CONTROL FIELD):

Not Applicable

F. RECORD LENGTH:

Not Applicable

G. BLOCKING FACTOR:

Not Applicable

H. PURPOSE:

The DBPL/I - DBPAC Interface (see Table 1) specifies the MFCB.STATEMENT.OPERATION code, DBPAC entry point name, and argument types and order for the various DBPL/I statments and substatements. Thus, it serves to specify for the DB preprocessor function (see Section IV, Topic B.1 of the DWB) what MFCB assignments and CALL statements are to be generated for each DBPL/I statement. Conversely, it specifies for DBPAC (see Sectiion IV, Topic B.2 of the DWB) what entry points will be entered and what and how information will be available at execution time for the performance of the various statement actions.

The various entry points and their argument types are declared by source code in SOURCE.LISRMAC member DBTEXT. Any program that includes the DB preprocessor also is given DBTEXT by an INCLUDE statement in DB.

TABLE 1.

	ROUTINE	OPTION	OPERATION	ENTRYPOINT	ARG-1	ARG-2
SS	CLOSE		00010000	DBPACFV	f	
SS	CLOSE	ERASE	00011000	DBPACFV	f	
SE	GET	FIELD	01100000	DBPACFV	f	v
SE	GET	INTERNAL FLD	01100100	DBPACFV	f	r
	GET	INDEX KEY	01100001	DBPACFV	f	v
	GET	KEY SET	01100000	DBPACFP	p	v
	GET	SUBFILE KEY SET	01010001	DBPACFP	f	p
	GET	LIST SET	01010000	DBPACFP	f	p
	GET	INDEX LIST SET	01010001	DBPACFP	f	p
	GET	SUBFILE LIST SET	0101001C	DBPACFP	f	p
	GET	RECORD	01000000	DBPACFR	f	r
	LOCATE		11010000	DBPACFV	f	v
	LOCATE	SUBFILE	11010010	DBPACFV	f	
SS	OPEN		00100000	DBPACFV	f	
SE	PUT	FIELD	10010000	DBPACFV	f	v
	READ	KEY	11100000	DBPACFV	f	v
	READ	KEY NOLOCK	11100100	DBPACFV	f	v
	READ	INDEX KEY	11100101	DBPACFV	f	v
	READ	SUBFILE KEY	11100010	DBPACFV	f	v
	READ	SUBFILE KEY NOLOCK	11100110	DBPACFV	f	v
	READ	PER SUBFILE	11101010	DBPACFV	f	v
	READ	PER SUBFILE NOLOCK	11101110	DBPACFV	f	
	READ	LIST	11101000	DBPACFP*	f	p
	READ	LIST NOLOCK	11101100	DBPACFP*	f	p
	READ	seq.	11110000	DBPACFV	f	
	READ	seq. NOLOCK	11110100	DBPACFV	f	
	READ	INDEX seq.	11110101	DBPACFV	f	
	READ	SUBFILE seq.	11110010	DBPACFV	f	
	READ	SUBFILE seq. NOLOCK	11110110	DBPACFV	f	
	READ	BACK	11111000	DBPACFV	f	
	READ	BACK NOLOCK	11111100	DBPACFV	f	
	READ	INDEX BACK	1111110C1	DBPACFV	f	
	READ	SUBFILE BACK	11111101	DBPACFV	f	
	READ	SUBFILE BACK NOLOCK	11111110	DBPACFV	f	
SE	REPUT	FIELD	10100000	DBPACFV	f	v
	UNLOCK		11000000	DBPACFV	f	
	UNLOCK	SUBFILE	11000010	DBPACFV	f	
	WRITE		10000000	DBPACFR	f	r

SS = substatement
SE = statement element
f = filename
p = list pointer
r = record work area
v = character string

*For READ LIST <NCLOCK> with the KEY (nn) clause, use entry point DBPACPF and a fullword subscript value as the third argument.

TOPIC B.3 - DATA BASE EXECUTIVE

A. DATA SET NAME:

DBPAC Error Codes

B. CREATED BY:

RDBPAC posts in MFCB.ERRCR.ONCODE.

C. TYPE OF FILE:

(4)Table

D. ORGANIZATION:

Sequential

E. KEY IDENTIFIER (CONTROL FIELD):

Error-code

F. RECORD LENGTH:

Not Applicable

G. BLOCKING FACTOR:

Not Applicable

H. PURPOSE:

The data base executive was written with the intent of handling all data base interfaces with the users of the NASIS system.

This is handled by the user writing a PL/I program and in it using the DEPL/I language extension to handle all of the input/output operations to the data base.

When the user's program is running, the data base executive will attempt to detect any and all errors which occur and communicate these errors back to the user's program.

The method of performing this communication of errors is through the use of the data base executive error codes. These are fixed binary numbers which have unique meanings and are transmitted back to the user's program using the MFCB (mainline file control block) as a vehicle of communication.

ERROR CODE	EXPLANATION OF ERROR
01	Illegal attempt to imply open.
03	Tried to imply an open on a new file-name without the use of an open command.
20	Trying to open a file when the header descriptor's DESCOK switch is off.
21	Trying to open for OUTPUT or UPDATE and the file is not the anchor or an associate or a descriptor file.
22	End-of list (READ LIST Statement).
23	Number of files exceeds the number allowed in the MFCB file array.
25	The user is not the owner of the file, but he is attempting an open for UPDATE or OUTPUT.
26	Attempted open of the file for INPUT, the DATA switch indicates no data.
27	Open attempted but its function was not INPUT, OUTPUT, or UPDATE.
28	Open issued for UPDATE or OUTPUT was prohibited by the MNTNING, MNTNABLE, or the DATA switch.
30	Operation code error.
31	Key field failed general validation (READ KEY or LOCATE).
32	Key field failed specific validation (READ KEY or LOCATE).
34	Erase attempted on CLOSE but the file is not open for UPDATE.
35	Erase attempted on descriptor file other than the anchor.
36	The GET FIELD operation attempted but the last record operation was a LOCATE.
38	The GET FIELD operation attempted but there is no current record.

39 GET RECORD operation attempted but the user is not the owner.

40 GET LIST attempted, but file is not inverted index.

41 Key is null (READ KEY or LOCATE).

42 Key sequence error (LOCATE sequential).

43 Duplicate key error (LOCATE direct).

44 Not an OUTPUT file for WRITE.

46 No current record (PUT or REPUT).

47 Current record not locked (PUT or REPUT).

49 PUT or REPUT to INPUT file.

50 REPUT to non-UPDATE file.

51 REPUT following LOCATE.

52 GET operation (field is not in descriptor table).

53 Field failed general validation (PUT or REPUT).

54 Field failed special validation (PUT or REPUT).

55 Null value to be PUT.

56 Bit field too long (PUT or REPUT).

57 PUT to non-null bit switch.

58 PUT to non-null fixed length field.

59 PUT to non-null variable length (single element) field.

61 Field would make record too long (PUT).

62 Field would make record too long (REPUT).

63 Element would make record too long (PUT).

64 Element would make record too long (REPUT).

65 Element field too long (PUT or REPUT).

66 Too many (variable) elements (PUT).

67 No GET before REPUT (variable elements).

68 No good GET before REPUT (variable elements).

69 Undefined field (PUT or REPUT).

70 REPUT to never PUT (null) field (record not found).

71 Too many (fixed) elements (PUT).

72 (Fixed) element would make record too long (PUT).

73 No GET before REPUT (fixed elements).

74 No good GET before REPUT (fixed elements).

75 Field too long (PUT or REPUT).

76 Key would make cross reference record too long (PUT or REPUT).

77 Cross reference not found on record.

78 Target field 'actual' length checking indicates truncation.

79 Command system trying to open someone elses STATIC or TRNSCT dataplexes for UPDATE or OUTPUT.

80 Command system opening a dataplex (other than STATIC or TRNSCT) for either OUTPUT or UPDATE.

83 GET KEY incompatible with list.

84 GET KEY sequence error.

85 Field length less than 2 found. Data Base damage.

86 Field length beyond reclen found. Data Base damage.

87 Field length not equal to 2 plus a multiple of element length found. Data Base damage.

88 Element length less than 1 found. Data Base damage.

89 Element length beyond field length found. Data Base damage.

90	Invalid DB2 header descriptor. DB1 descriptor or damage.	
91	Field descriptor reclen less than 78. Descriptor damage.	
92	Field length less than 2 in descriptor. Descriptor damage.	
93	Field length beyond reclen in descriptor. Descriptor damage.	
94	VALIDARG longer than 50 bytes would be truncated. Descriptor damage.	
95	SECURITY field length invalid. Should be 2 plus a multiple of 8.	
96	No descriptor found for key field. Descriptor damage.	
97	Invalid field length in index record found. Data base damage.	
98	Record missing from index region. Data base damage.	
99	End of data.	(VISAM)
104	Keys equal - sequence error.	(VISAM+100)
108	Key not found.	(VISAM+100)
112	Keys out of sequence.	(VISAM+100)
115	Keys do not coincide.	(VISAM+100)
120	Keys coincide.	(VISAM+100)
124	Invalid retrieval address.	(VISAM+100)
128	Invalid record length.	(VISAM+100)
131	Position past end of data set.	(VISAM+100)
136	Position before start of data set.	(VISAM+100)
140	Exceed maximum number of overflow pages.	(VISAM+100)
144	Exceed maximum size of shared data set.	(VISAM+100)
145	No data set-name found which is like the given	

one. (VISAM)

200 Attempt to (PUT or REPUT) null pattern.

201 Attempt to (PUT or REPUT) to readonly field.

202 Undefined subfile or indexed field (READ, LOCATE, or UNLOCK).

203 Not an indexed field.

204 Not a subfile <control> field.

205 LOCATE to INPUT file.

206 No current anchor record (LOCATE SUBFILE).

207 Anchor record not locked (LOCATE SUBFILE).

208 No current subrecord (READ PER SUBFILE).

209 Anchor record not current (delete subrecord).

210 The file is not open (to post FLDTAB).

211 Descriptor damage detected while posting FLDTAB.

212 Anchor record not locked (delete subrecord).

213 Anchor record not parent of subrecord (delete subrecord).

214 Subrecord id not found in control field (delete subrecord).

215 Duplicate fixed length element (PUT or REPUT).

216 Duplicate varying length element (PUT or REPUT).

217 LOCATE SUBFILE not done because 131071 regions used.

218 NAMEFLD field length invalid. Should be 2 plus a multiple of 9.

219 GET superfield requires current subfile record.

220 RSECTYCD field length invalid. Should be 2 plus a multiple of 9.

221 Non-owner attempted to open associate having record level security.

TOPIC B.4 - DATA BASE EXECUTIVE

A. DATA SET NAME:

Mainline File Control Block

B. CREATED BY:

Declared by DB Preprocessor Function.

C. TYPE OF FILE:

(4) Table

D. ORGANIZATION:

Linear structure followed by an array of linear structures.

E. KEY IDENTIFIER (CONTROL FIELD):

Within DBPAC the mainline file control block is a parameter named MFCB. Outside DBPAC each mainline file control block is an independent external controlled structure whose name is the DBPL/I file name (PLEX in the retrieval system). For this reason, file names must not conflict with other external name system. This file name is not padded with dollar signs the way a file title must be. The file name is passed as an argument in CALL statements to DBPAC and thus becomes the MFCB parameter.

F. RECORD LENGTH:

1324 bytes (hexadecimal 52C)

This is the length of the whole control block including the necessary dope vectors and a thirty-seven element array allowing up to thirty-seven data sets in a data base. The number of elements in the array may be adjusted, if necessary: - the control block size will be adjusted by 24 bytes per element and the MFCB.FILE.ARR_SIZE field must be suitably initialized but no changes are necessary in DBPAC or in other MFCB's.

G. BLOCKING FACTOR:

Not Applicable

H. PURPOSE:

The MFCB control block is used for communication between mainline programs and DBPAC. For DBPL/I statements in the mainline, the DB preprocessor function generates statements that post fields in the MFCB , such as the operation code. At execution time, the posted MFCB is passed as an argument to DBPAC. DBPAC performs the operation indicated in the MFCB, making reference to other fields in the MFCB as necessary and posting fields in the MFCB, such as MFCB.ERROR.ONCODE, which may subsequently be referenced in the mainline.

I. PL/I DECLARATION:

```
DECLARE
1 MFCB,                          /*MAINLINE FILE CONTROL BLOCK*/
  2 INITIALIZED BIT(2),          /*00: NEVER INITIALIZED      */
                                 /*10: INITIALIZED, CLOSED    */
                                 /*11: INITIALIZED, OPENED    */
  2 FILLER_1 BIT(6),             /*NOT USED                   */
  2 STATEMENT,                   /*CR FUNCTION                */
    3 OPERATION BIT(8),          /*CODE                       */
    3 ONFIELD CHAR(8),           /*FIELD NAME                 */
  2 FILLER_2 CHAR(3),            /*NOT USED                   */
  2 ERROR,
    3 SYSTEM BIT(1),             /*1: STANDARD DBPAC ACTION   */
                                 /*0: USER ERROR ROUTINE      */
    3 FILLER_3 BIT(7),           /*NOT USED                   */
    3 ONCODE FIXED BINARY,
    3 ROUTINE LABEL,             /*USER'S                     */
    3 ONRETURN LABEL,            /*IN MAINLINE                */
  2 FILE,
    3 ONFILE CHAR(8),            /*FILE TITLE                 */
    3 OLFILE CHAR(8),            /*TO SAVE FILE TITLE IF DYNAM*/
    3 OWNER_ID CHAR(8),          /*OWNER OF THE FILE          */
    3 DSNAME CHAR(35),           /*DATA SET NAME              */
    3 ATTRIBUTES
      4 ACCESS BIT(1),           /*0: DIRECT                  */
                                 /*1: SEQUENTIAL              */
      4 SAVE_FUNC BIT(2),        /*TO SAVE FUNCTION IF DYNAMIC*/
      4 FILLER_4 BIT(3),         /*NOT USED                   */
      4 FUNCTION BIT(2),         /*10: INPUT                  */
                                 /*01: OUTPUT                 */
                                 /*11: UPDATE                 */
    3 CURRENT_FILE FIXED BINARY,/*SUBSCRIPT IN FILE.ARRAY*/
    3 LAST_FILE FIXED BINARY,
              /*NUMBER OF FILES IN FILEPLEX OR DATAPLEX*/
    3 ARR_SIZE FIXED BINARY(15),/*DECLARED ARRAY SIZE    */
    3 ARRAY (37),
      4 FILE_NAME CHAR(8),
      4 DTP POINTER,             /*DESCRIPTOR TABLE ADDRESS   */
```

```
  4 FCBP POINTER,          /*FILE CONTROL BLOCK ADDRESS */
  4 KYC FIXED BINARY(15),/*SUBSCRIPT OF KEY FIELD     */
                           /*DESCRIPTOR IS ALWAYS =1.   */
  4 SWITCHES,
    5 CURRENT BIT(1),
    5 LOCKED BIT(1),
    5 WRITE BIT(1),        /*FORCE WRITE                */
    5 REWRITE BIT(1),      /*FORCE REWRITE              */
    5 ABSENT BIT(1),       /*NULL OR SECURED RECORD     */
    5 OPENED BIT(1),       /*THE FILE IS OPEN           */
    5 FILLER_5 BIT(2),     /*NOT USED                   */
  4 FILLER_6 CHAR(1),      /*NOT USED                   */
  4 RECORDCT FIXED BINARY(15);/* # OF USABLE DESC.      */
```

J. DETAIL NOTES:

INITIALIZED - used entirely within DBPAC.

STATEMENT.OPERATION - see Section III, Topic B.2 of the DWB for the codes that are posted here by the DB preprocessor function.

STATEMENT.ONFIELD - posted by the DB preprocessor function.

ERROR.SYSTEM - posted by the DB preprocessor function.

ERROR.ONCODE - posted by DBPAC when an error is detected but not reset for successful operations. See Section III, Topic B.3 of the DWB for the DBPAC Error Codes.

ERROR.ROUTINE - posted by the DB preprocessor function.

ERROR.ONRETURN - posted by DBPAC when an error is detected.

FILE.ONFILE - posted by the DB preprocessor function. When the first character is not a pound sign indicating a descriptor file, DBPAC shifts the value one character to the right and posts a leading black character.

FILE.OLFILE - used within DBPAC to detect need for reinitialization.

FILE.OWNER_ID - used within DBPAC.

FILE.DSNAME - used within DBPAC.

FILE.ATTRIBUTES.ACCESS and FUNCTION - posted by either

the DE preprocessor function or, when in default, by DBPAC.

FILE.SAVE_FUNC - used within DBPAC.

FILE.CURRENT_FILE - used within DBPAC.

FILE.LAST_FILE - used within DBPAC.

FILE.ARR_SIZE - set by the DB preprocessor function indicating the dimension of the FILE.ARRAY.

FILE.ARRAY - this array is used within DBPAC. Each element of the array is a linear structure of fields relating to a data set. When the mainline is accessing a descriptor region or an inverted index, only the first element is used. Otherwise, the first element relates to the anchor data set and subsequent elements relate to associated and subfile and inverted index data sets.

FILE.ARRAY.FILE_NAME - the "title" of the data set having a leading blank or pound sign and a trailing blank or suffix.

FILE.ARRAY.DTP - the address of the (dynamically allocated) descriptor table for this data set.

FILE.ARRAY.FCBP - the address of the (dynamically allocated) file control block for this data set.

FILE.ARRAY.KYC - the subscript of the key field descriptor in the descriptor table array is always 1.

FILE.ARRAY.KYC(1) - one (the anchor) plus the number of associate data sets.

FILE.ARRAY.KYC(2) - KYC(1) plus the number of subfile data sets.

FILE.ARRAY.SWITCHES - switches used by DBPAC for the status of the data set.

FILE.ARRAY.RECORDCT - the number of descriptors in the descriptor table array. This number does not include descriptors of fields a given user does not have field security clearance to access.

TOPIC B.5 - DATA BASE EXECUTIVE

A. DATA SET NAME:

List Structure

B. CREATED BY:

RDBPAC and RDBLIST

C. TYPE OF FILE:

(4) Table

D. ORGANIZATION:

Structure containing and adjustable array

E. KEY IDENTIFIER (CONTROL FIELD):

Not Applicable

F. RECORD LENGTH:

40 byte prefix plus number-of-keys times internal key length

G. BLOCKING FACTOR:

Not Applicable

H. PURPOSE:

The list structure includes the primary list of keys and all supporting information. The list segments created by the data base executive are all linked together by the PREV pointer and the NEXT pointer.

I. PL/I DECLARATION:

```
DECLARE
 1 LIST BASED(LIST_PTR),          /*MAIN STORAGE KEY LIST*/
   3 PREFIX,
     5 CHAIN,                    /*CONNECTING SEGMENTS  */
       7 PTR,                    /*NULL:  END OF CHAIN  */
         9 PREV PTR,             /*BACKWARDS CHAIN      */
         9 NEXT PTR,             /*FORWARDS CHAIN       */
       7 CONTINUED BIT(1),       /*NEXT SEGMENT IS      */
                                 /*CONT. OF THIS ONE?   */
     5 SPARE BIT(31),            /*NOT USED             */
     5 STRING_SIZE FIXED BIN,    /*ONLY POSTED BY ALLOC.*/
     5 KEY,
```

```
    7 MAX_COUNT FIXED BIN,      /*COUNT LIMT THIS SEGMT*/
    7 COUNT FIXED BIN,          /*KEYS PER THIS SEGMT  */
    7 CURSOR,                   /*FOR THIS SEGMT       */
      9 READ FIXED BIN,         /*FOR READ PER LIST    */
      9 GET FIXED BIN,          /*FOR GET KEY          */
    7 SIZE FIXED BIN,           /*BYTES PER KEY        */
    7 FIELD_NAME CHAR(8),       /*KEY|CONTRL FLD NAME  */
                                /*KEYS APPLY TO        */
    7 CONVERSION CHAR(8),       /*RTN NAME FOR OUTPUT  */
                                /*NULL_VALUE:  NONE    */
3 KEYS CHAR(IIST_STRING_SIZE    /*CHAR(KEY.SIZE *      */
  REFER(LIST.STRING_SIZE));     /*MAX_COUNT)           */
```

TOPIC B.6 - DATA BASE EXECUTIVE

A. DATA SET NAME:

LISTERR - List Error Control Block

B. CREATED BY:

Allocated by RDBMTT.

Error fields are posted by DB preprocessor and RDBLIST. List chain anchors are initialized by RDBMTT and posted by RDBPAC and RDBLIST.

C. TYPE OF FILE:

(4) Table

D. ORGANIZATION:

Simple structure

E. KEY IDENTIFIER (CONTROL FIELD):

Not applicable

F. RECORD LENGTH:

68 bytes (20 bytes data + 48 bytes PL/I dope vectors, etc.)

G. BLOCKING FACTOR:

Not applicable

H. PURPOSE:

The list error control block holds the list segment chain anchors. (The chain used by the FREE all LISTs statement.) It also is the point of communication between RDBLIST and mainline programs for list error handling when no MFCB is involved in the operation. (When an MFCB is involved in a list error situation, the MFCB is used for error indication, etc.)

I. PL/I DECLARATION:

```
DECLARE

1 LISTERR CTL EXT,              /*COMMON CONTROL BLOCK      */
  3 ERROR,
    5 SYSTEM BIT(1)             /*1: SYSTEM ACTION          */
```

```
   INIT('1'B),                  /*0: GC TO USER ERROR RTNE  */
  5 ONCODE FIXED BIN(15)        /*1: INVALID LIST OPERATION */
    INIT(0),                    /*2: INCOMPATIBLE LISTS     */
                                /* GET LIST KEY SET ERRORS: */
                                /*4: NULL INPUT LIST        */
                                /*5: NO GET KEY SINCE RESET */
                                /*6: INCOMPATIBLE LISTS     */
                                /* GET LIST KEY INTO ERROR: */
                                /*7: KEY SEQUENCE ERROR     */
                                /*8: TRUNC, TARGET TOO SHORT*/
                                /* SET LIST LIKE LIST ERROR:*/
                                /*9: INVALID SIZE           */
                                /* PUT LIST KEY FROM ERRORS:*/
                                /*10: NULL TARGET LIST      */
                                /*11: WRONG LENGTH KEY VALUE*/
                                /*12: KEY SEQUENCE ERROR    */
  5 ROUTINE LABEL               /*USER ERROR RTNE ADDRESS   */
3 PTR,                          /*NULL: NO CHAIN            */
                                /*ALLOCATOR MUST INITIALIZE */
  5(FIRST,                      /*FORWARD CHAIN ANCHOR      */
    LAST) PTR;                  /*BACKWARD CHAIN ANCHOR     */
```

TOPIC B.7 - DATA BASE EXECUTIVE

A. DATA SET NAME:

Data base Descriptor File,

for example:

SAOWNER.ASRDI$.ASRDI$#

where:

"ownerid" is the 1-8 character TSS identification of the owner of the data base.

"data base" is the 6 character data base name with the dollar sign character used for padding.

B. CREATED BY:

RDBEDIT - the Descriptor Editor program

C. TYPE OF FILE:

(6) Data Base Descriptor file

D. ORGANIZATION

VISAM - Virtual Indexed Sequential Access Method (TSS) - organized in one or more regions of contiguous records; one region for the anchor data set descriptors and, as necessary, a region for each associate, subfile and inverted index dataset's descriptors.

Records have the varying length universal record format (URF) built by DBPAC.

E. KEY IDENTIFIER (CONTROL FIELD):

The VISAM key length is 15 bytes consisting of:

7 character region name and 8 character FLDNAME.

A DBPL/I descriptor FILE is a contiguous set of records having the same region name. The DBPL/I FILE shall be OPENed using an 8 character TITLE value consisting of:

1 pound sign character (#) (signifying descriptor file).

6 character data base name with the dollar sign

character ($) used for padding.

1 suffix character.

The suffix character shall be from the following ranges:

blank	anchor file descriptors
1-9	associate file descriptors
Z-Q	subfile descriptors
A-P	inverted index descriptors

DBPAC uses the data base name and suffix to automatically generate the region name value for the keys.

The DBPL/I KEY value is only the 8 character FLDNAME (name of the field being described) that completes the VISAM key value.

F. RECORD LENGTH:

34 bytes minimum for a file descriptor record.
78 bytes minimum for a field descriptor record.

G. BLOCKING FACTORS:

One 4096 byte page (block) will hold about 40 average descriptors; enough for a few regions for a simple data base. For a complicated data base with many data sets, fields, secondary fields, and security codes, three or more pages (blocks) may be required.

H. PURPOSE:

A data base descriptor file describes a data base in terms of the datasets, records, and fields the data base is composed of, and indicates their interrelationships. A data base descriptor file is created and maintained or modified by RDBEDIT, the descriptor editor, which is a system service program.

Then the data base may be loaded, maintained, retrieved, etc. by programs using RDBPAC for data base access. RDBPAC, when OPENing a data base, reads the descriptor file and from it builds a table that governs its further actions.

I. SAMPLE DATA BASE:

A sample data base is used to illustrate in detail how the descriptors shall describe a complicated data base. See Figure 1.

The record layout in Figure 2 shows all eight record layouts in the sample data base for reference in the following sections of this specification.

J. DESCRIPTOR REGIONS:

The sample descriptor file consists of eight regions (having the suffixes ' ',1,Z,Y,A,B,C,D) corresponding to the eight data sets in the sample data base. (Of course VISAM alphabetizes them ' ',A,B,C,D,Y,Z,1.)

Each descriptor region has at least one file descriptor record and two field descriptor records (key and RECLEN).

Dummy descriptor records are required in anchor regions when the data base has associate and/or subfile datasets. They are also required in associate regions when subfile(s) are controlled from the associate dataset.

Figure 3 tabulates all the file, field, and dummy descriptors required to describe the sample data base.

K. FILE DESCRIPTOR RECORD:

The file descriptor record describes the dataset as a whole. It is uniquely identified by having a key (FLDNAME) of 8 blanks. It has the field values shown below. See Figure 4 for the field values of the sample descriptor file. All values should be posted except that if RECSECFP is NULL, RSECTYCD does not apply.

FIELD	VALUE	COMMENTS
FLDNAME	8 blanks	DBPL/I key
DESCOK	OFF ON	incomplete descriptors. descriptors are complete.
FILETYPE	ANCHOR ASSOCIATE SUBFILE INDEX	type of data set being described
DESCRCT	numeric	number of field descriptors in this region. (The file descriptor is not to be counted.)
BSELNGTH	numeric	length in bytes of fixed portion

		of records including RECLEN, key and fixed primary fields. For a spanned index, this includes the key suffix byte.
SPANNED	OFF	ordinary records.
	ON	spanned records with internally suffixed keys.
DATA	OFF	no data on file yet.
	ON	retrieval is possible.
MNTNABLE	ON	maintenance is allowed.
	OFF	maintenance is prohibited.
MNTNING	OFF	no maintenance is in progress.
	ON	maintenance is in progress.
LOADABLE	ON	loading is allowed.
	OFF	loading this data set is prohibited.
RECSECFP	null	records do not have a record security field.
	numeric	offset in bytes of record security field in records.
RSECTYCD	9 byte elements	zero or more record level security codes.
	8 alphameric	record security password.
	1 byte	mask for comparison with record security field.

L. FIELD DESCRIPTOR RECORDS:

A field descriptor record indicates that a particular named field occurs in the dataset being described. It is uniquely identified within the region by having a key (FLDNAME) that is the name of the field. There are two kinds of field descriptors:

primary direct
secondary (direct and indirect)

All field descriptor records may have the values shown below:

FIELD	VALUE	COMMENTS
FLDNAME	8 alphamerics blank padded	unique field name. DBPL/I key.

GENERCRT	8 alphamerics blank padded	name of generic routine for testing input values.
VALIDRTN	8 alphamerics blank padded	name of routine for testing and/or converting input values.
VALIDARG	0-50 bytes	argument to be supplied to VALIDRTN
NUMALIGN	OFF	string alignment, left justification
	ON	numeric alignment, right justification
REFORMAT	8 alphamerics blank padded	name of routine for converting output values.
SECURITY	8 alphamerics asterisk padded	0-18 field security passwords

GENERCRT, VALIDRTN, VALIDARG, and NUMALIGN may even be posted for secondary (read only) fields because linear search, for example, may have to transform values to be used as comparands.

M. PRIMARY DIRECT FIELD DESCRIPTOR RECORDS:

A primary direct field descriptor record describes a maintainable field that occurs on each record of the dataset being described. There are five kinds of primary fields:

single fixed bit
single fixed byte
single varying byte
multiple fixed byte
multiple varying byte

In addition to the field values shown in Section L, all primary descriptors have READONLY OFF and a selection of the following field values.

FIELD	VALUE	COMMENTS
READONLY	OFF	field value may be maintained (PUT and REPUT.)
VARFLD	FIXED	fixed length field in fixed portion of records. portion of records.

BITFLD	OFF	byte field
	ON	bit field
FLDPOSIT	numeric	if VARFLD is FIXED, offset in bytes of field. if VARFLD is VARYING, relative field in variable portion of records.
FLDLEN	numeric	if BITFLD is ON, offset of bit (0,2,4 or 6) in byte specified by FLDPOSIT. if VARFLD is FIXED, internal length of field in bytes. if VARFLD is VARYING, maximum internal length of field in bytes with internal field length prefix.
ELTLIM	0	field does not have elements.
	numeric	maximum number of elements to be PUT into field or, for a control field, maximum number of sub-records per parent record.
ELTLEN	numeric	if VARELT is FIXED, internal length of elements in bytes. if VARELT is VARYING, maximum internal length of elements in bytes with internal element length prefix.
VARELT	FIXED	fixed length elements.
	VARYING	varying length elements.
UNIQUELT	OFF	duplicate element values are allowed.
	ON	internal element values must be unique.
INVFILE	alphabetic	descriptor region suffix for inverted index dataset. (null if none.)
INDEXEXT	OFF	index internal values of field.
	ON	index external values of field. (External length may require index key length greater than internal length.)

A single fixed bit field descriptor has:

VARFLD	FIXED
BITFLD	ON

FLDPOSIT	offset in bytes
FLDLEN	offset in bits (0,2,4 or 6)
INVFILE	null (may not be indexed)

See VEHAIRCD in the sample data base.

A single fixed byte field descriptor has:

VARFLD	FIXED
BITFLD	OFF
FLDPOSIT	offset in bytes
FLDLEN	internal length in bytes
INVFILE	optional if FLDLEN less than 254

See EMPPAYCI, EMPINSCL and VEHMAKE in the sample data base.

A single varying byte field descriptor has:

VARFLD	VARYING
FLDPOSIT	relative varying field
FLDLEN	maximum internal length including internal 2 byte field length prefix
ELTLIM	0 (zero)
INVFILE	optional if (FLDLEN-2) less than 254

See KIDNAME in the sample data base.

A multiple fixed byte field descriptor has:

VARFLD	VARYING
FLDPOSIT	relative varying field
FLDLEN	maximum internal field length including internal 2 byte field length prefix
ELTLIM	maximum number of elements
ELTLEN	internal element length in bytes
VARELT	FIXED
UNIQUELT	optional
INVFILE	optional if ELTLEN less than 254

See the EMPKID and EMPVEH control fields in the sample data base.

A multiple varying byte field descriptor has:

VARFLD	VARYING
FLDPOSIT	relative varying field
FLDLEN	maximum internal field length including internal 2 byte field length prefix and internal 1 byte element length prefixes
ELTLIM	maximum number of elements
ELTLEN	maximum internal element length including internal 1 byte element length

	prefix. VARELT VARYING
UNIQUELT	optional
INVFILE	optional if (ELTLEN-1) less than 254.

See the KIDPET field inthe sample data base.

CONTROL FIELD DESCRIPTORS

Every descriptor region must have a key descriptor for the field that uniquely identifies records in a dataset. It is a primary direct field descriptor record for a single fixed byte field.

SECURITY	must be null.
READONLY	is OFF
VARFLD	is FIXED
BITFLD	is OFF
FLDPOSIT	is 4
INVFILE	must be null

Each associate key descriptor is identical (except the region suffix) to the anchor key discriptor. See the EMPNAME field in the sample data base.

Each subfile dataset in a data base requires:

1. a control field in the anchor or an associate dataset
2. a separate descriptor region for the subfile dataset containging:

2a. file descriptor record
2b. RECLEN secondary descriptor record
2c. subrecord id key descriptor record
2d. parent key secondary descriptor record
2e. descriptor records for other subrecord fields.

1. A subfile control field descriptor describes a secondary multiple fixed byte field maintained by RDBPAC.

FLDNAME	is a six-character name suffixed by two blanks applying to the subfile.
GENERCRT	is DBCVTID
VALIDRTN	is null
NUMALIGN	is ON
REFORMAT	is DBFMTID
SECURITY	is optional
READONLY	is ON
VARFLD	is VARYING
FLDPOSIT	is relative varying field

FLDLEN	is maximum internal field length
ELTLIM	is maximum number of elements. Note that FLDLEN or ELTLIM limits the maximum number of subrecords per parent record.
ELTLEN	is 3
VAREIT	is FIXED
UNIQUEIT	is ON
SUBCNTRL	is CN
SUBFILE	is alphabetic character descriptor region suffix for subfile dataset.
INVFILE	must be null

See EMPKID and EMPVEH in the sample data base.

2a. A subfile file descriptor record has:

FILETYPE	SUBFILE

2b. A subrecord RECLEN descriptor is standard.

2c. A subrecord id key descriptor describes a primary single fixed byte field:

FLDNAME	is the six-character subfile name suffixed by "ID".
GENERCRT	is DBCVTID
VALIDRTN	is null
NUMALIGN	is CN
REFORMAT	is DBFMTID
SECURITY	must be null
READONLY	is OFF
VARFLD	is FIXED
BITFLD	is OFF
FLDPOSIT	is 4
INVFILE	must be null

See EMPKIDID and EMPVEHID in the sample data base.

2d. A subrecord parent key descriptor describes a secondary single fixed byte field.

FLDNAME	is the six-character subfile name suffixed by "PK".
GENERCRT, VALIDRTN, NUMALIGN, REFORMAT and FLDLEN	are the same as the anchor key descriptor.
SECURITY	is optional
READONLY	is ON
VARFLD	is FIXED
BITFLD	is OFF
FLDPOSIT	is 7
INVFILE	must be null

See EMPKIDPK and EMPVEHPK in the sample data base.

2e. Record level security may be independently specified for the anchor, associate and/or subfile datasets. It may not be specified for an inverted index dataset. Each dataset to have record level security must have:

RECSECFP	field position of record security field
RSECTYCD	optional

in its file descriptor record and a primary direct field descriptor record for a single fixed byte field as shown in Figure 5 and as follows:

FLDNAME	RECSEC suffixed by the descriptor region suffix (blnak for the anchor and a blank.
GENERCRT	DBCVTHX
VALIDARG	null
NUMALIGN	OFF
REFORMAT	DBFMTHX
SECUIRTY	optional
READONLY	OFF
VARFLD	FIXED
BITFLD	OFF
FLDPOSIT	on anchor or associate datasets, after the key (ie. 4 + key FLDLEN + 1). on subfile datasets, after the parent key field (ie. 4 + 3 + parent key FLDLEN + 1).
INVFILE	optional

See the RECSEC, RECSEC1, EMPKIDRS and EMPVEHRS fields in the sample data base.

N. SECONDARY READONLY FIELD DESCRIPTOR RECORDS:

A secondary field descriptor record describes a derived field made up of one or more component fields. There are two types: direct and indirect.

A direct secondary field descriptor redescribes part of all of one primary field or a field automatically maintained by RDBPAC such as RECLEN and subfile control and parent key fields. A direct secondary descriptor has the same field values as a primary descriptor with the following qualifications:

RFADONLY is always ON. Field may only be retrieved (GET).

INVFILE must be null.

FLDPOSIT and FLDLEN will specify an internal location within or equal to a primary field.

Otherwise, the descriptor fields may specify a direct secondary field like any of the five types of primary fields. A secondary of the same type of length provides renaming and perhaps an alternate REFORMAT. A secondary "single fixed byte" with a shorter FLDLNGTH provides for subfields. A secondary "single varying byte" redefining a primary "multiple fixed byte" obtains the concatenation of the internal element values.

Every descriptor region shall have a secondary direct field descriptor for a single fixed byte RECLEN as follows:

FLDNAME	is RECLEN
GENERCRT	is DBCVTRL
VALIDRTN	is null
NUMALIGN	is ON
REFORMAT	is DBFMTRL
SECURITY	is optional
READONLY	is ON
VARFLD	is FIXED
BITFLD	is OFF
FLDPOSIT	is 0 (zero)
FLDLEN	is 4
INVFILE	must be null

(No dummy descriptors are used for RECLEN. Direct secondary descriptors may be specified by the Data Base Analyst to provide unique field names for the various RECLENs in a data base.)

An indirect secondary field descriptor describes a "superfield" make up of one or more primary or direct secondary component fields. No more than one of the component fields may be a multi-element field. The component field values will be concatenated in NAMEFLD order for retrieval. (If there is a multi-element component field, then the superfield will yield multiple values.) An indirect secondary descriptor has the field values shown below.

FIELD	VALUE	COMMENTS
READONLY	ON	field may only be retrieved
NAMEFLD	9 byte	one to 16 component field

elements	specifications
hex '00'	use external value of component
hex '80'	use internal value of component
followed by 8 alphmerics blank padded	primary or direct secondary component fieldname

REFORMAT 8 alphamerics name of routine for converting
blank padded concatenated output value

If the component fields specified in NAMEFLD are all from the same dataset (anchor, associate or subfile), then the indirect secondary descriptor goes in the descriptor region for that dataset. (Dummy descriptor(s) are required for the indirect secondary descriptor if it is on an associate or subfile.) See the KIDID field in the sample data base.

If the componenets are from an associate file and from a subfile controlled from that associate file, then the indirect secondary descriptor goes in the descriptor region for that associate.

Otherwise the indirect secondary descriptor goes in the anchor descriptor region and no dummy descriptors are required. See the EMPTYPE field in the sample data base.

If any component is from a subfile dataset, the (1.) no components may be from any other subfile dataset and (2.) to GET such a field, a mainline program must first obtain a current record in the subfile -- RDBPAC will automatically ensure that the parent and associate record(s) are available when required.

Every anchor and associate descriptor region shall have a secondary indirect field descriptor for the key field, see Figure 6.

FLDNAME	is 'FILEKEY '
READONLY	is ON
NAMEFLD	has one element consisting of hex '00' followed by the name of the primary key field.
REFORMAT	is NULL.

(No dummy descriptor is used for FILEKEY on associate datasets.)

O. DUMMY DESCRIPTOR RECORDS

A dummy field descriptor indicates that the field occurs in an asscciate or subfile data set and if it has an inverted index data set. It has the field values shown below.

FIELD	VALUE	COMMENTS
FLDNAME	8 alphamerics blank padded	field name. DBPL/I key.
ASSOCFIL	numeric character	descriptor region suffix for associate data set. (Null if none.)
SUBFILE	alphabetic character	descriptor region suffix for subfile data set. (Null if none.)
SUBCNTRL	CFF	dummy descriptor for subfile field.
	ON	primary (dummy if ASSOCFIL is posted) descriptor for subfile control field.
INVFILE	alphabetic character	descriptor region suffix for inverted index data set. (Null if none.)

The descriptor file, shown graphically in Figure 7 and 8, is so designed that the anchor region describes the anchor records and also has dummy descriptors for all other fields cn associate (1) or subfile (Z,Y) records thus indicating the presence of all associate and subfile data sets. It also indicates the presence of all inverted indexes for the data base (A,B,C,D).

An associate region (1) describes a data set of associate records and has dummy descriptors for all other fields on subfiles (Y) depending on the associate record. It also indicates the presence of all inverted indexes for the associate and dependent subfiles (C,D).

A subfile region (Z) describes a data set of subfile records and indicates the presence of all inverted indexes for the subrecords (B).

An index region (A) describes a data set cf inverted index records.

This all enables DBPAC to access a whole data base, an associate portion of a data base, a subfile, or an

inverted index as if it were a degenerate case of a data base.

P. INVERTED INDEX DESCRIPTORS:

If INVFILE is posted for a primary direct field, then a separate descriptor region must exist for the inverted index dataset. It contains only the following:

FILE DESCRIPTOR RECORD

FILETYPE is INDEX
SPANNED is optional
BSELNGTH is 4 + index key FLDLEN (+1 if SPANNED).

RECLEN SECONDARY DIRECT FIELD DESCRIPTOR RECORD

Single Fixed Byte

INDEX KEY SECONDARY DIRECT FIELD DESCRIPTOR RECORD

Single Fixed Byte

FLDNAME is same as indexed FLDNAME
READONLY is ON
FLDPOSIT is 4

If indexed field descriptor has INDEXEXT OFF, the FLDLEN is maximum internal indexed field value length (without 2 byte internal field length or 1 byte internal element length) and REFORMAT is same as indexed field REFORMAT. If indexed field descriptor has INDEXEXT ON, then FLDLEN is maximum external indexed field value length and REFORMAT is null or a blank stripper. In either case, FLDLEN does not include the internal "sequence number" suffix if SPANNED.

CROSS REFERENCES SECONDARY DIRECT FIELD DESCRIPTOR

Record Multiple Fixed Byte

FLDNAME is same as indexed field's record key FLDNAME.
READONLY is ON.
FLDPOSIT is 1.
FLDLEN is 4000.
ELTLIM is 4000.
ELTLEN is same as indexed field's record key FLDLEN.
REFORMAT is same as indexed field's record key REFORMAT.

Q. FILE AND FIELD DESCRIPTOR RECORD FORMATS:

File Descriptor Record Format

1	RECLEN	0-3	Fixed	Binary	Length of header record, in bytes, including itself.
2	KEY	4-18	Fixed	EBCDIC	Identifier for this descriptor. Contains file name .
3	FLENAME	4-10	Fixed	EBCDIC	Seven-character file name for this descriptor. Contains data base and suffix.
4	DATAPLEX	4-9	Fixed	EBCDIC	Dataplex name padded with $s to 6 characters.
5	SUFFIX	10	Fixed	EBCDIC	Identifier dataset.
6	FLDNAME	11-18	Fixed	EBCDIC	Contains blanks.
7	FILETYPE	19	Fixed	EBCDIC	1: Anchor 2: Associate 3: Subfile 4: Inverted index
8	DESCRCT	20-21	Fixed	Binary	Number of field descriptors for this dataset.
9	BSELNGTH	22-23	Fixed	Binary	Length of fixed portion of record.
10	DESCOK	24.0	Fixed	Bit	0: incomplete descriptors. 1: complete descriptors.
11	SPANNED	24.2	Fixed	Bit	Applicable if FILETYPE=4: 0: ordinary records 1: spanned records with internally suffixed keys.
12	DATA	24.4	Fixed	Bit	0: No data on file. 1: Retrieval is possible.

13	MNTNABLE	24.6	Fixed	Bit	0: Maintenance prohibited. 1: Maintenance allowed.
14	MNTNING	25.0	Fixed	Bit	0: Maintenance not in progress. 1: Maintenance in progress.
15	--------	25.2	Fixed	Bit	Currently not applicable-Null. 1: Check record security.
16	LOADABLE	25.4	Fixed	Bit	0: Loading prohibited. 1: Loading allowed.
17	--------	25.6	Fixed	Bit	Currently not applicable-Null.
18	REMAINS	26-29	Fixed	------	Currently not applicable-Null.
19	RECSECFP	30-31	Fixed	Binary	Record security field offset in records; null if none.
20	RSECTYCD	VrFld1	Var	------	0-18 record security specifications consisting of a NASIS-id padded with $s to 8 characters and a one byte mask.

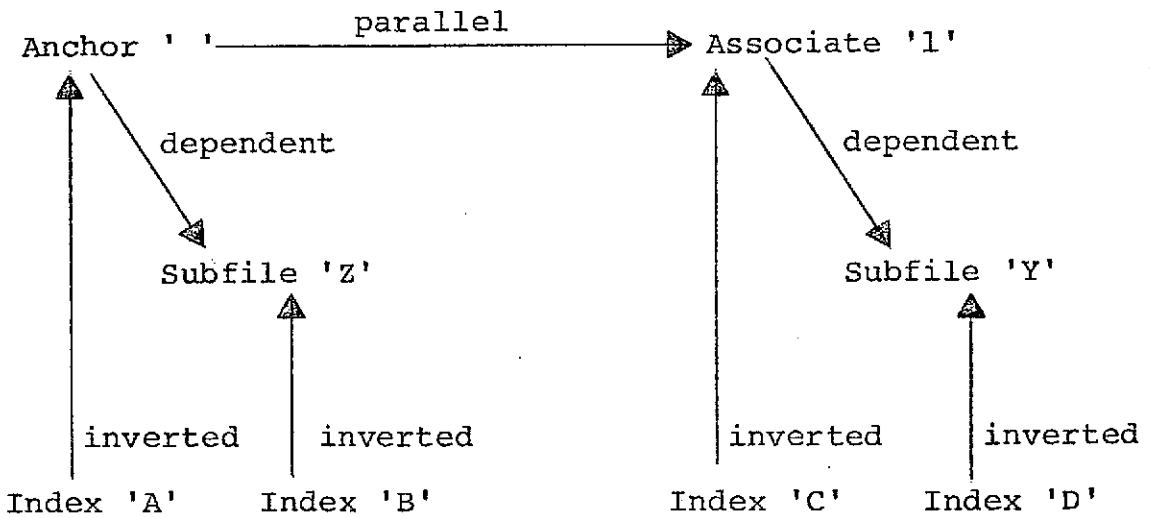
Field Descriptor Record Format

1	RECLEN	0-3	Fixed	Binary	Length of entire descriptor, in bytes, including itself.
2	KEY	4-18	Fixed	EBCDIC	Identifier for this descriptor. Contains file and field names.
3	FLENAME	4-10	Fixed	EBCDIC	Seven character file name for these descriptors.
4	FLDNAME	11-18	Fixed	EBCDIC	Name of field within file.
5	ASSOCFIL	19	Fixed	EBCDIC	Suffix of associated linear file which contains this field; null if none.
6	SUBFILE	20	Fixed	EBCDIC	Suffix of subfile which contains this field or which is controlled by this field; null if none.
7	INVFILE	21	Fixed	EBCDIC	Suffix of inverted file which indexes this field; null if none.
8	READONLY	22.0	Fixed	Bit	0: (Re)Put allowed. 1: (Re)Put prohibited.
9	SUBCNTRL	22.2	Fixed	Bit	Applicable if SUBFILE is non-null: 0: Field is on subfile. 1: This is control field.
10	VARFLD	22.4	Fixed	Bit	0: Fixed length field. 1: Varying length field.
11	BITFLD	22.6	Fixed	Bit	Applicable if VARFLD=0 0: Byte field. 1: Bit field.

12	NUMALIGN	23.0	Fixed	Bit	0: String (left) align. 1: Numeric (right) align.
13	VARELT	23.2	Fixed	Bit	Applicable if ELTLIM>0: 0: Fixed length elements. 1: Varying length elements.
14	UNIQUELT	23.4	Fixed	Bit	Applicable if ELTLIM>0: 0: Duplicate elements allowed. 1: Duplicate elements prohibited.
15	INDEXEXT	23.6	Fixed	Bit	Applicable if INVFILE is non-null. 0: Index internal values. 1: Index external values.
16	GENERCRT	24-31	Fixed	EBCDIC	Name of routine to be used for testing type of input characters (numeric, alpha, etc.);null if none.
17	VALIDRIN	32-39	Fixed	EBCDIC	Name of routine to be used for special validation or conversion of input data; null if none. Uses argument in VALIDARG.
18	REFORMAT	40-47	Fixed	EBCDIC	Name of routine to be used for any necessary output reformatting or conversion; null if none.
19	SPARE	48-55	Fixed	------	Currently not applicable-Null.
20	NAMECNT	56-57	Fixed	------	Currently not applicable-Null.
21	FLDPOSIT	58-59	Fixed	Binary	If VARFLD=0: byte

					offset in record. If VARFLD=1: relative varying field.
22	FLDLEN	60-61	Fixed	Binary	If BITFLD=1: bit offset in byte. If VARFLD=0: field length in bytes. If VARFLD=1: maximum field length in bytes including 2 byte length indicator.
23	DFLDLEN	62-63	Fixed	------	Currently not applicable-Null.
24	ELTLIM	64-65	Fixed	Binary	Applicable if VARFLD=1: 0: not multi-element. >0: maximum number of elements allowed.
25	DELTLIM	66-67	Fixed	------	Currently not applicable-Null.
26	ELTLEN	68-69	Fixed	Binary	If VARELT=0: element length in bytes. If VARELT=1: maximum element length in bytes including 1 byte length indicator.
27	DELTLEN	70-71	Fixed	------	Currently not applicable-Null.
28	VALIDARG	VrFld1	VarHex		Argument to be used with VALIDRTN (test mask, limit, etc.). Fifty bytes maximum. Null if none.
29	NAMEFLD	VrFld2	Var		0-18 Superfield components consisting of a one byte function code (80x: external field element, 00x: internal field element) and an 8 character component field name.

30	SECURITY	VrFld3	Var	EBCDIC	0-18 field security codes consisting of a NASIS-id padded with *s to 8 characters.



(Note: A simple data plex consists of only an anchor data set. A more complicated dataplex than the sample may have multiple index, associate, and/or subfile datasets, but the principles for describing it are shown in the sample.)

FIGURE 1. SAMPLE DATAPLEX

BYTES
8 – double word
4 – word
2 – halfword
1 – 2 packed-decimal digits

IBM System/360 Record Layout Worksheet

INTERNATIONAL BUSINESS MACHINES CORPORATION

GX20-1711 0 U/M 025
Printed in U.S.A. (Rept. 4/70)

Record Name SAMPLE DATAPLEX 02/07/72

APPLICATION

Page ____ of ____
Date

ANCHOR

RECLEN EMPNAME (KEY) RECSEC EMPPAYCL EMPKID (CONTROL FIELD)

CHARACTERISTICS ▶ B H FL

HEXADECIMAL
DECIMAL

ASSOCIATE '1'

RECLEN EMPNAME (KEY) RECSEC1 EMPINSCL EMPVEH (CONTROL FIELD)

CHR ▶ H FL

HEX
DEC

HEX	DEC
00	0
100	256
200	512
300	768
400	1024
500	1280
600	1536
700	1792
800	2048
900	2304
A00	2560
B00	2816
C00	3072
D00	3328
E00	3584
F00	3840

SUBFILE 'Z'

RECLEN EMPKIDI (KEY) EMPKIDPK (PARENT KEY) EMPKIDR KIDNAME KIDPET

CHR ▶ B H FL H FL

HEX
DEC

SUBFILE 'Y'

RECLEN EMPVEHI (KEY) EMPVENPK (PARENT KEY) EMPVEHR VEHAIRC VENMAKE (CODE)

CHR ▶ B B

HEX
DEC

INDEX 'A'

RECLEN EMPAYCL (KEY) (SUFFIX) EMPNAME (CROSS REFERENCES)

CHARACTERISTICS ▶ H FL

HEXADECIMAL
DECIMAL

INDEX 'B'

RECLEN KIDNAME (KEY) EMPKIDID (CROSS REFERENCES)

CHR ▶ H FL

HEX
DEC

INDEX 'C'

RECLEN EMPINSCL (KEY) EMPNAME (CROSS REFERENCES)

CHR ▶ H FL

HEX
DEC

INDEX 'D'

RECLEN VEHMAKE (KEY) (EXTERNAL FORM) EMPVEHID (CROSS REFERENCES)

CHR ▶ H FL

HEX
DEC

Fold to here

CHARACTERISTIC CODES ▶

C – character, 8-bit code
X – hexadecimal, 4-bit code
B – binary
F – fixed-point, full word
H – fixed-point, halfword
E – floating-point, full word
D – floating-point, double word
P – packed decimal
Z – zoned decimal
A – address value, full word
Y – address value, halfword
V – address, external symbol
S – address, base displacement

FIGURE 2 – SAMPLE DATAPLEX RECORD LAYOUT

FOLDOUT FRAME 2

FOLDOUT FRAME 1

	Anchor	Associate	Subfile	Subfile
Region:	'PLEX$$ '	'PLEX$$1'	'PLEX$$Z'	'PLEX$$Y'
	file descriptor	file descriptor	file descriptor	file descriptor
Field descriptors:	RECLEN	RECLEN	RECLEN	RECLEN
	EMPNAME (Key)	EMPNAME (Key)		
	FILEKEY	FILEKEY		
	RECSEC			
	EMPTYPE			
	EMPPAYCL			
	EMPKID (control)			
	EMPKIDID		EMPKIDID (Key)	
	EMPKIDPK		EMPKIDPK	
	EMPKIDRS		EMPKIDRS	
	KIDNAME		KIDNAME	
	KIDPET		KIDPET	
	KIDID		KIDID	
	RECSEC1	RECSEC1		
	EMPINSCL	EMPINSCL		
	EMPVEH	EMPVEH (control)		
	EMPVEHID	EMPVEHID		EMPVEHID (Key)
	EMPVEHPK	EMPVEHPK		EMPVEHPK
	EMPVEHRS	EMPVEHRS		EMPVEHRS
	VEHAIRCD	VEHAIRCD		VEHAIRCD
	VEHMAKE	VEHMAKE		VEHMAKE
	dummy descriptors			

	Index	Index	Index	Index
Region:	'PLEX$$A'	'PLEX$$B'	'PLEX$$C'	'PLEX$$D'
	file descriptor	file descriptor	file descriptor	file descriptor
Field descriptors:	RECLEN	RECLEN	RECLEN	RECLEN
Key:	EMPPAYCL	KIDNAME	EMPINSCL	VEHMAKE
Cross references:	EMPNAME	EMPKIDID	EMPNAME	EMPVEHID

FIGURE 3. SAMPLE DATAPLEX DESCRIPTOR LIST

REGION	DESCOK	FILETYPE	DESCRCT	BSELNGTH	SPANNED DATA	MNTNABLE	MNTNING	LOADABLE	RECSECFP
PLEX$$	ON	ANCHOR	21	17	OFF	OFF	ON	OFF	ON	14
PLEX$$1	ON	ASSOCIATE	11	20	OFF	OFF	ON	OFF	ON	14
PLEX$$Z	ON	SUBFILE	7	18	OFF	OFF	ON	OFF	ON	17
PLEX$$Y	ON	SUBFILE	6	21	OFF	OFF	ON	OFF	ON	17
PLEX$$A	ON	INDEX	3	7	ON	OFF	ON	OFF	ON	null
PLEX$$B	ON	INDEX	3	14	OFF	OFF	ON	OFF	ON	null
PLEX$$C	ON	INDEX	3	9	OFF	OFF	ON	OFF	ON	null
PLEX$$D	ON	INDEX	3	14	OFF	OFF	ON	OFF	ON	null

FIGURE 4. SAMPLE FILE DESCRIPTORS

SAMPLE PRIMARY DIRECT FIELD DESCRIPTORS

FLDNAME	SECURITY	READONLY	VARFLD=FIXED/VARYING	BITFLD	FLDPOSIT	FLDLEN	NUMALIGN	ELTLIM	ELTLEN	VARELT=FIXED/VARYING	UNIQUELT	GENERCRT	VALIDRTN	VALIDARG	REFORMAT	INVFILE	INDEXEXT
PLEX$$ EMPNAME	X	OFF	F	OFF	4	10	OFF					( )	( )	( )	( )	X	
PLEX$$ RECSEC	( )	OFF	F	OFF	14	1	OFF					DBCYTHX	X	X	DBFMTHX		
PLEX$$ EMPPAYCL	( )	OFF	F	OFF	15	2	OFF					( )	( )	( )	( )	A	OFF
PLEX$$1EMPNAME	X	OFF	F	OFF	4	10	*					*	*	*	*	X	
PLEX$$1RECSEC1	( )	OFF	F	OFF	14	1	OFF					DBCVTHX	X	X	DBFMTHX		
PLEX$$1EMPINSCL	( )	OFF	F	OFF	15	5	OFF					( )	( )	( )	( )	C	OFF
PLEX$$ZEMPKIDID	X	OFF	F	OFF	4	3	ON					DBCVTID	X	X	DBFMTID	X	
PLEX$$ZEMPKIDRS	( )	OFF	F	OFF	17	1	OFF					DBCVTHX	X	X	DBFMTHX		
PLEX$$ZKIDNAME	( )	OFF	V			1	10	OFF					( )	( )	( )	( )	B	OFF
PLEX$$ZKIDPET	( )	OFF	V			2	40	OFF	5	10	V	OFF	( )	( )	( )	( )	( )	( )
PLEX$$YEMPVEHID	X	OFF	F	OFF	4	3	ON					DBCVTID	X	X	DBFMTID	X	
PLEX$$YEMPVEHRS	( )	OFF	F	OFF	17	1	OFF					DBCVTHX	X	X	DBFMTHX		
PLEX$$YVEHAIRCD	( )	OFF	F	ON	18	0	OFF					( )	( )	( )	( )	X	
PLEX$$YVEHMAKE	( )	OFF	F	OFF	19	2	OFF					( )	( )	( )	( )	D	ON

SAMPLE SECONDARY DIRECT FIELD DESCRIPTORS

																	SUBCNTRL	SUBFILE
PLEX$$ RECLEN	( )	ON	F	OFF	0	4	ON					DBCVTRL	X	X	DBFMTRL	X	X	X
PLEX$$ EMPKID	( )	ON	V		1	4000	ON	20	3	F	ON	DBCVTID	X	X	DBFMTID	X	ON	Z
PLEX$$1RECLEN	( )	ON	F	OFF	0	4	ON					DBCVTRL	X	X	DBFMTRL	X	X	X
PLEX$$1EMPVEH	( )	ON	V		1	4000	ON	5	3	F	ON	DBCVTID	X	X	DBFMTID	X	ON	Y
PLEX$$ZRECLEN	( )	ON	F	OFF	0	4	ON					DBCVTRL	X	X	DBFMTRL	X	X	X
PLEX$$ZEMPKIDPK	( )	ON	F	OFF	7	10	*					*	*	*	*	X		
PLEX$$YRECLEN	( )	ON	F	OFF	0	4	ON					DBCVTRL	X	X	DBFMTRL	X	X	X
PLEX$$YEMPVEHPK	( )	ON	F	OFF	7	10	*					*	*	*	*	X		

FIGURE 5. SAMPLE DIRECT FIELD DESCRIPTORS

FLDNAME	SECURITY	READONLY	NAMEFLD	REFORMAT
PLEX$$ FILEKEY	X	ON	(EMPNAME)	X
PLEX$$ EMPTYPE	( )	ON	(EMPPAYCL,EMPINSCL)	( )
PLEX$$1FILEKEY	X	ON	(EMPNAME)	X
PLEX$$ZKIDID	( )	ON	(KIDNAME,EMPKIDPK)	( )

FIGURE 6. SAMPLE INDIRECT SECONDARY FIELD DESCRIPTORS

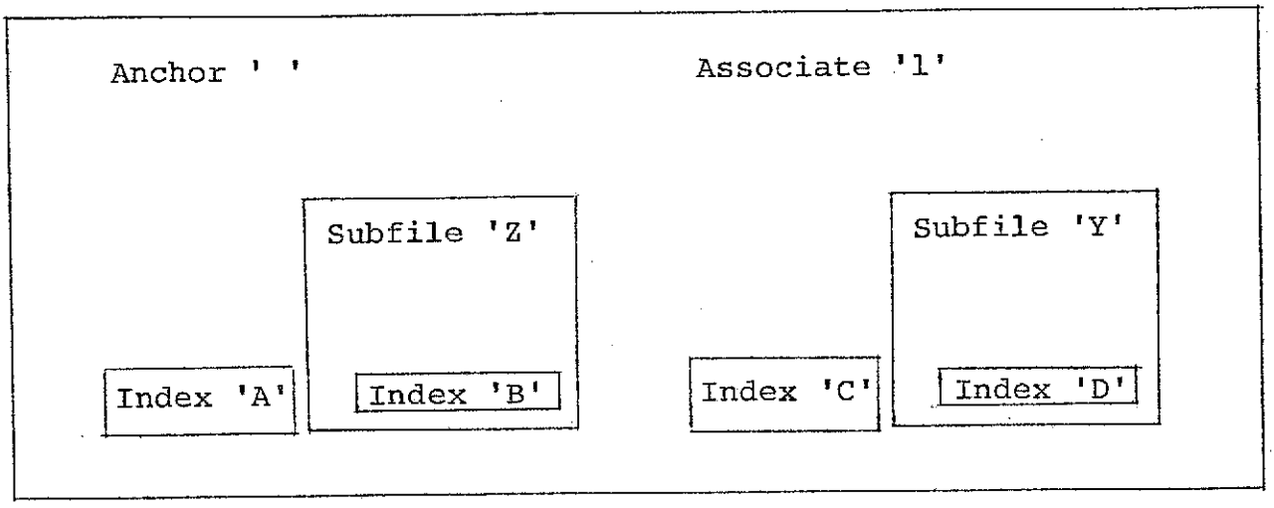


FIGURE 7. SAMPLE DESCRIPTOR FILE

	FLDNAME	ASSOCFIL	SUBFILE	SUBCNTRL	INVFILE
PLEX$$	EMPKIDID		Z	OFF	
"	EMPKIDPK		Z	OFF	
"	EMPKIDRS		Z	OFF	
"	KIDNAME		Z	OFF	B
"	KIDPET		Z	OFF	
"	KIDID		Z	OFF	
"	RECSEC1	1			
"	EMPINSCL	1			C
"	EMPVEH	1	Y	ON	
"	EMPVEHID	1	Y	OFF	
"	EMPVEHPK	1	Y	OFF	
"	EMPVEHRS	1	Y	OFF	
"	VEHAIRCD	1	Y	OFF	
"	VEHMAKE	1	Y	OFF	D
PLEX$$1	EMPVEHID		Y	OFF	
"	EMPVEHPK		Y	OFF	
"	EMPVEHRS		Y	OFF	
"	VEHAIRCD		Y	OFF	
"	VEHMAKE		Y	OFF	D

FIGURE 8. SAMPLE DUMMY FIELD DESCRIPTORS

TOPIC B.8 - DATA BASE EXECUTIVE

A. DATA SET NAME:

DBPL/I - DBLIST Interface

B. CREATED BY:

DB Preprocessor Function

C. TYPE OF FILE:

(4) Table

D. ORGANIZATION:

Documentary Table

E. KEY IDENTIFIER (CONTROL FIELD):

Not Applicable

F. RECORD LENGTH:

Not Applicable

G. BLOCKING FACTOR:

Not Applicable

H. PURPOSE:

The DBPL/I - DBLIST Interface (see Table 1) specifies the DBLIST entry point name and the argument types and order for the various DBPL/I statements. Thus, it serves to specify for the DB Preprocessor function (see Section IV, Topic B.5) what CALL statements are to be generated for each DBPI/I statement. Conversely, it specifies for DBLIST (see Section IV, Topic B.5) what entry points will be entered and what and how information will be available at execution time for the performance of the various statement actions.

The various entry points and their argument types are declared by source code in SOURCER.LISRMAC member DBTEXT. Any program that includes the DB preprocessor also is given DBTEXT by an INCLUDE statement in DB.

TABLE 1.

DBPL/I	GENERATED PL/I
FREE LIST;	CALL DBPAC;
FREE LIST(p1,p2);	CALL DBPACP(p1);
GET LIST(p1) KEY(0);	CALL DBGLK0(p1);
GET LIST(p1) KEY INTO (st);	CALL DBGLKI(p1,st);
GET LIST(p1) INTERNAL KEY INTO (st);	CALL DBGLIK(p1,st);
GET LIST(p1) KEY(n) INTO(st);	CALL DBGLKN(p1,n,st);
GET LIST(p1) KEY SET(p2);	CALL DBGLKS(p1,p2);
SET LIST(p2) SIZE(n) LIKE LIST(p1);	CALL DBSLLL(p2,n,p1);

where:

p1,p2 are POINTER
n is FIXED BINARY(31)
st is CHARACTER(-)VARYING

TOPIC C.1 - UTILITIES

A. DATA SET NAME:

NASIS USERIDS

B. CREATED BY:

CREATIDS (procdef)

C. TYPE OF FILE:

VISAM

D. ORGANIZATION:

Variable format

E. KEY IDENTIFIER (CONTROL FIELD):

8 Character NASISID, key length 8, key position 4.

F. RECORD LENGTH:

4000 bytes

G. BLOCKING FACTOR:

Not Applicable

H. PURPOSE:

Maintain for each NASIS ID, the corresponding password, timeslice, user authority and list of valid files.

TOPIC C.2 - UTILITIES

A. DATA SET NAME:

JOINIDS

B. CREATED BY:

USERJOIN

C. TYPE OF FILE:

VISAM

D. ORGANIZATION:

Variable format

E. KEY IDENTIFIER (CONTROL FIELD):

A JOINed TSS-ID, key length 8, key position 4.

F. RECORD LENGTH:

30 bytes.

G. BLOCKING FACTOR:

Not Applicable

H. PURPOSE:

This is the list of JOINed TSS-IDs for the program MERGE.

TOPIC C.3 - UTILITIES

A. DATA SET NAME:

EDIT.LISRMLF

B. CREATED BY:

Not Applicable

C. TYPE OF FILE:

VISAM

D. ORGANIZATION:

Index Sequential

E. KEY IDENTIFIER (CONTROL FIELD):

The fifteen byte key is composed of the eight byte message key concatenated to the seven byte line number.

F. RECORD LENGTH:

V(132)

G. BLOCKING FACTOR:

Not Applicable

H. PURPOSE:

This data set contains a copy of the NASIS system messages used by the various modules for prompting and diagnostic messages. After editing it, the DBA will use this file to replace the current system message file, LISRLIB(0)(LISRMLF).

TOPIC D.1 - MAINTENANCE

A. DATA SET NAME:

RDBLOAD ERROR_CODES Table

B. CREATED BY:

RDBLOAD

C. TYPE OF FILE:

Core table

D. ORGANIZATION:

Sequential array

E. KEY IDENTIFIER (CONTROL FIELD):

Not Applicable

F. RECORD LENGTH:

Defined as (0:216) character (1)

G. BLOCKING FACTOR:

Not Applicable

H. PURPOSE:

This table contains codes to be used to control the action taken for each DBPAC error that may occur.

TOPIC D.2 - MAINTENANCE

A. DATA SET NAME:

TRNSCT Data Set Descriptors

B. CREATED BY:

CORRECT (RECORR)

C. TYPE OF FILE:

VI (Indexed) - Anchor

D. ORGANIZATION:

Indexed Sequential (VISAM)

E. KEY IDENTIFIER (CONTROL FIELD):

Offset of 4, fixed field (255)

F. RECORD LENGTH:

4000 / V

G. BLOCKING FACTOR:

Not Applicable

H. PURPOSE:

The purpose of the transaction data base is to serve as a temporary repository for changes to one of the database datafiles, until the datafile owner can validate the change and execute the maintenance program to apply it.

The TRNSCT data base will consist of fields defined as follows:

KEY - This will be an alphanumeric Fixed field 255 bytes in length consisting of the data base name (padded to six characters with $) concatenated with the OWNER-ID, (padded to eight characters with *) of the affected data base concatenated with the anchor's key. The last fourteen (14) bytes of the key will be a time stamp field.

NASISID - This will be an alphanumeric Fixed field, 8 bytes in length, and will contain the

NASIS-ID of where the transaction was created.

OPCODE - This is an alphanumeric Fixed field, 3 bytes in length, which indicates the operation to be performed.

FIELD - This is an alphanumeric Fixed field, 8 bytes in length, which indicates the field of a data base which is to be updated.

START - For field context operations, this field will contain the starting location of the context. It will be an alphanumeric Fixed field 4 bytes in length.

END - For field context operations, this field will contain the ending location of the context. It will be an alphanumeric Fixed field, 4 bytes in length.

OLDDATA - A field which will contain data. This will be the old data field in the instance of the field context.

NEWDATA - A field which will contain data. This will be the new data field in the instance of the field context.

SUBKEY - This is an alphanumeric fixed field, 10 bytes in length, which indicates the subfile key to be corrected.

SUBCTL - This is an alphanumeric fixed field, 8 bytes in length, which is the subfile control field for subfile being corrected.

One important consideration is that conversion, validation and reformatting routines can be written and used to check the data as it enters the data base, thus causing many errors to be detected before maintenance is ever run.

TOPIC D.3 - MAINTENANCE

A. DATA SET NAME:

CORRECT Data Display Format

B. CREATED BY:

CORRECT (RDBCORR)

C. TYPE OF FILE:

Terminal Display

D. ORGANIZATION:

Not Applicable

E. KEY IDENTIFIER (CONTROL FIELD):

Not Applicable

F. RECORD LENGTH:

Not Applicable

G. BLOCKING FACTOR:

Not Applicable

H. PURPOSE:

The purpose of this display is to present as much of the data contained in the field as possible to the user.

```
CORRECT XXXXXXXX, XXXXXXXXXXXXXX

(LENGTH = 80, ELEMENTS = 4)

E001  :XXXXXXXX
E002  :XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 002   XXXXXX
E003  :XXXXXXXX
E004  :XXXXXXXX
```

TOPIC D.4 - MAINTENANCE

A. DATA SET NAME:

RDBLOAD Error Data Set

B. CREATED BY:

Not Applicable

C. TYPE OF FILE:

VISAM or VSAM

D. ORGANIZATION:

Sequential

E. KEY IDENTIFIER (CONTROL FIELD):

(Must be the same as that of the RDBLOAD input data set).

F. RECORD LENGTH:

(Must be the same as that of the RDBLOAD input data set).

G. BLOCKING FACTOR:

Not Applicable

H. PURPOSE:

This data set serves as the repository for all input records that cannot be successfully loaded.

TOPIC D.5 - MAINTENANCE

A. DATA SET NAME:

Database Inverted Index Format

B. CREATED BY:

A descriptor region is created by RDBEDIT (the descriptor editor) for each inverted index. It generally consists of a header record and three field descriptor records.

Inverted index records are originally written by either RINVRTSS (the sort method of inversion) or RDBINVRT (an inversion utility program) or RDBPAC (inverting concurrently with database loading).

C. TYPE OF FILE:

(6) Database File

D. ORGANIZATION:

TSS VISAM - Virtual Indexed Sequential Access Method

E. KEY IDENTIFIER (CONTROL FIELD):

The index key field name is the same as the indexed field name, e.g. AUTHOR, SUBJTERM, KEYWORD etc. The VISAM key length is the maximum (internal) length of the indexed values (plus 1 for a spanned index) and may not exceed 255 bytes.

F. RECORD LENGTH:

Variable - 4000 bytes maximum.

G. BLOCKING FACTOR:

VISAM blocks records into pages of 4096 bytes.

H. PURPOSE:

The purpose of the inverted index files is to give the system a fast, efficient method of storing and accessing the list of records in a database file that contain a particular data element value.

The records of inverted files consist of a universal record form with a specialized structure. The structure consists of the concatenation of the

following fields:

the VISAM record length field (RECLEN)
4 bytes, fixed length, binary

the VISAM key field
1-254 bytes, fixed length, indexed field element value
optionally suffixed by
1 byte, fixed length, binary record number within region if the index is SPANNED.

the cross references field
2 bytes, fixed length, binary field length
followed by one or more fixed length
anchor or subfile internal key values in ascending collating sequence.

In a SPANNED index, the first record of a region has key suffix zero, and possible continuation records (up to 255) in a continuous 'region'. All records in a region, except possibly the last, are maximum length (have the maximum number of whole cross references).

TOPIC D.6 - MAINTENANCE

A. DATA SET NAME:

Descriptor Editor Data Display Format

B. CREATED BY:

Descriptor Editor DISPLAY Command (RDBEDDP)

C. TYPE OF FILE:

Terminal Display

D. ORGANIZATION:

Not Applicable

E. KEY IDENTIFIER (CONTROL FIELD):

Not Applicable

F. RECORD LENGTH:

Not Applicable

G. BLOCKING FACTOR:

Not Applicable

H. PURPOSE:

The purpose of this display is to allow the user of the Descriptor Editor to review the specifications for a particular field descriptor.

I. SAMPLE DISPLAY:

```
FIELD NAME.............xxxxxxxx
FIELD TYPE.............xx
ALIGNMENT..............x
FIELD FORMAT...........xx
FIELD LENGTH...........xxxx
ELEMENT LENGTH.........xxx
ELEMENT NUMBER.........xxxx
UNIQUE ELEMENTS........x
CONVERSION ROUTINE.....xxxxxxxx
FORMATTING ROUTINE.....xxxxxxxx
VALIDATION ROUTINE.....xxxxxxxx
VALIDATION ARGUMENT....xxxxxxxxxxxxx
          xxxxxxxxxxxxxxxxxxxxxxxxxxxx
          xxxxxxxxxxxxxxxxxxxxxxxxxxxx
```

```
          XXXXXXXXXXXXXXXXXXXXXXXXXXXX
INDEX FILE ID.............X
INDEX KEY FORM............X
EXTERNAL KEY LENGTH.......X
INDEX SPANNED.............X
ASSOCIATE FILE ID.........X
SUBFILE CONTROL FIELD....X
SUBFILE ID................X
BASE FIELD NAME...........XXXXXXXX
BASE FIELD OFFSET.........XXXX
SUPERFIELD COMPONENTS....X.XXXXXXXX
          X.XXXXXXXX      X.XXXXXXXX
          X.XXXXXXXX      X.XXXX XXXX
          X.XXXXXXXX      X.XXXXXXXX
          X.XXXXXXXX      X.XXXXXXXX
          X.XXXXXXXX      X.XXXXXXXX
          X.XXXXXXXX      X.XXXXXXXX
          X.XXXXXXXX      X.XXXXXXXX
          X.XXXXXXXX
SECURITY..................XXXXXXXX
          XXXXXXXX     XXXXXXXX
          XXXXXXXX     XXXXXXXX
          XXXXXXXX     XXXXXXXX
          XXXXXXXX     XXXXXXXX
          XXXXXXXX     XXXXXXXX
          XXXXXXXX     XXXXXXXX
          XXXXXXXX     XXXXXXXX
          XXXXXXXX     XXXXXXXX
          XXXXXXXX
```

TOPIC D.7 - MAINTENANCE

A. DATA SET NAME:

Descriptor Editor Field Name Display Format

B. CREATED BY;

Descriptor Editor (FIELDS) Command (RDBEDFD)

C. TYPE OF FIIE:

Terminal Display

D. ORGANIZATICN:

Not Applicable

E. KEY IDENTIFIER (CCNTROL FIEID):

Not Applicable

F. RECORD LENGTH:

Not Applicable

H. PURPOSE:

The purpose of this display is to allow the user of the Descriptor Editor to review the names of all of the fields described thus far in CREATE mode. In UPDATE mode the user is presented a list of the descriptor descriptor field names.

I. SAMPLE DISPLAY:

```
 XXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXX     XXXXXXXX     XXXXXXXX
XXXXXXXX     XXXXXXXX     XXXXXXXX
XXXXXXXX     XXXXXXXX
```

NOTE: A Total of 57 names can be displayed on one screen.

TOPIC D.8 - MAINTENANCE

A. DATA SET NAME:

RDBLOAD Input Data Set

B. CREATED BY:

Not Applicable

C. TYPE OF FILE:

VISAM or VSAM

D. ORGANIZATICN:

Sequential

E. KEY IDENTIFIER (CONTROL FIELD):

Usually same as that of the data base to be loaded.

F. RECORD LENGTH:

Not Applicable

G. BLOCKING FACTOR:

Not Applicable

H. PURPOSE:

This data set serves as the source of input records for RDBLOAD.

TOPIC D.9 - MAINTENANCE

A. DATA SET NAME:

Descriptor Editor Listing Format

B. CREATED BY:

Descriptor Editor Print Command (RDBEDPR)

C. TYPE OF FILE:

1403 Printer Display

D. ORGANIZATION:

Not Applicable

E. KEY IDENTIFIER (CONTROL FIELD):

Not Applicable

F. RECORD LENGTH:

133

G. BLOCKING FACTOR:

Not Applicable

H. PURPOSE:

The purpose of this display is to list the contents of each descriptor field and each file descriptor in character form.

I. SAMPLE DISPLAY:

See Figure 1

TOPIC D.10 - MAINTENANCE

A. DATA SET NAME:

INVERT Restart File

'INVERT.PARM.'||FILENAME -
where FILENAME is the six character dataplex name.

B. CREATED BY:

RDBSIVRT mcdule.

C. TYPE OF FIIE:

Sequential

D. ORGANIZATICN:

VSAM

E. KEY IDENTIFIER (CONTROL FIELD):

Not Applicable

F. RECORD LENGTH:

255 bytes (Variable)

G. BLOCKING FACTOR:

Not Applicable

H. PURPOSE:

This file provides a restart key for the first phase of DBSIVRT.

TOPIC D.11 - MAINTENANCE

A. DATA SET NAME:

INVERT SORTIN File

'SORTIN.'||FILENAME||'.'||FIELD

1. FILENAME is the six character data base name.

2. FIELD is the 1-8 character field name that is being inverted.

B. CREATED BY:

First step of DBSIVRT.

C. TYPE OF FILE:

Sequential

D. ORGANIZATION:

VSAM

E. KEY IDENTIFIER (CONTROL FIELD):

First field is the maximum length value of the field being inverted.

F. RECORD LENGTH:

4000 bytes (Variable). Record consists of maximum length value of field being inverted concatenated with file Key.

G. BLOCKING FACTOR:

Not Applicable

H. PURPOSE:

This file is the input to the second step, which is the TSS sort step of DBSIVRT.

TOPIC D.12 - MAINTENANCE

A. DATA SET NAME:

INVERT SORTOUT File

'SORTOUT.'||FILENAME||'.'||FIELD

1. FILENAME is the six character data base name.

2. FIELD is the 1-8 character field name that is being inverted.

B. CREATED BY:

Sort step of DBSIVRT.

C. TYPE OF FILE:

Sequential

D. ORGANIZATION:

VSAM

E. KEY IDENTIFIER (CONTROL FIELD):

First field is the maximum length value of the field being inverted.

F. RECORD LENGTH:

4000 bytes (Variable). Record consists of maximum length value of field being inverted concatenated with the file Key.

G. BLOCKING FACTOR:

Not Applicable

H. PURPOSE:

This file is the input to step three of DBSIVRT.

TOPIC D.13 - MAINTENANCE

A. DATA SET NAME:

INVERT PLEX File

'PLEX.'||FILENAME||'.'||FIELD

1. FILENAME is the six character dataplex name.

2. FIELD is the 1-8 character field name that is being inverted.

B. CREATED BY:

Step three of DBSIVRT.

C. TYPE OF FILE:

Indexed Sequential

D. ORGANIZATION:

VISAM

E. KEY IDENTIFIER (CONTROL FIELD):

Key of file is internal field value being inverted concatenated with blanks up to the maximum external field length. If index file is spanned, span character is concatenated as last position of Key.

F. RECORD LENGTH:

4000 bytes (Variable). Record is identical in format as an index file record.

G. BLOCKING FACTOR:

Not Applicable

H. PURPOSE:

This file is the input to the last step, translation step, of DBSIVRT.

TOPIC D.14 - MAINTENANCE

A. DATA SET NAME:

INVERT RANGE File

'RANGE.'||FILENAME||'.'||FIELD

1. FILENAME is the six character data base name.

2. FIELD is the 1-8 character field name that is being inverted.

B. CREATED BY:

Step three of IBSIVRT if not indexed external, if indexed external, step four creates this data set.

C. TYPE OF FILE:

Indexed Sequential

D. ORGANIZATION:

VISAM

E. KEY IDENTIFIER (CONTROL FIELD):

Key of the file is the maximum length value of the field being inverted. If index file is spanned, span character is concatenated as last position of Key.

F. RECORD LENGTH:

4000 bytes (Variable). Record is identical to index file record.

G. BLOCKING FACTOR:

Not Applicable

H. PURPOSE:

This file is the input to the combine module for index updates.

TOPIC D.15 - MAINTENANCE

A. DATA SET NAME:

DESCRP.CHKPOINT

B. CREATED BY:

Descriptor Editor Checkpoint (RDBEDCP)

C. TYPE OF FILE:

TSS VAM

D. ORGANIZATION:

Sequential

E. KEY IDENTIFIER (CONTROL FIELD):

Not Applicable

F. RECORD LENGTH:

The records are of varying length of which the maximum is dynamically determined at execution time. The maximum possible value is 4000.

G. BLOCKING FACTOR:

Not Applicable

H. PURPOSE:

The purpose of this dataset is for storing sufficient information from the descriptor tables so that the user can continue creating the descriptor file at a future time through use of the RESTORE command.

The first record consists of those items from the X structure whose value must be preserved. The second record consists of the entire content of the FIELD structure. The next group of records will contain the field descriptor information. There will be one record for each existing field, consisting of the information in the appropriate FLD structure concatenated with the information contained in the appropriate SECURITY, SUPER, and VALID structures where applicable. Following the field descriptor records are records containing the header descriptor information, one for each existing file. These records consist of the information from the appropriate HDR structure

concatenated to the information from the proper RECSEC structure when applicable.

TOPIC D.16 - MAINTENANCE

A. DATA SET NAME:

MERGE INDEX File

'INDMRG.PARM.'//FILENAME - FILENAME is the six character data base name.

B. CREATED BY:

RDBINDM MODULE

C. TYPE OF FILE:

SEQUENTIAL

D. ORGANIZATION:

VSAM

E. KEY IDENTIFIER (CONTROL FIELD):

Not Applicable

F. RECORD LENGTH:

255 Bytes (variable)

G. BLOCKING FACTOR:

Not Applicable

H. PURPOSE:

This file provides a restart key for restart of DBINDM.

TOPIC D.17 - MAINTENANCE

A. DATA SET NAME:

Descriptor Editor REVIEW Display Format

B. CREATED BY:

Descriptor Editor REVIEW Command (RDBEDRV)

C. TYPE OF FILE:

Terminal Display

D. ORGANIZATION:

Not Applicable

E. KEY IDENTIFIER (CONTROL FIELD):

Not Applicable

F. RECORD LENGTH:

Not Applicable

G. BLOCKING FACTOR:

Not Applicable

H. PURPOSE:

The purpose of this display is to allow the user of the Descriptor Editor to review the exact contents of any descriptor record in any descriptor region.

I. SAMPLE DISPLAYS:

```
FILE DESCRIPTOR
---------------
SUFFIX  =x            FILETYPE=x
DESCRCT =xxxxx        BSELNGTH=xxxxx
DESCOK  =x            SPANNED =x
DATA    =x            MNTNABLE=x
MNTNING =x            LCADABLE=x
RECSECFP=xxxxx        REMAINS =xxxxxxxx
RSECTYCD=xxxxxxxx:xx    xxxxxxxx:xx
         xxxxxxxx:xx    xxxxxxxx:xx
         xxxxxxxx:xx    xxxxxxxx:xx
         xxxxxxxx:xx    xxxxxxxx:xx
         xxxxxxxx:xx    xxxxxxxx:xx
         xxxxxxxx:xx    xxxxxxxx:xx
         xxxxxxxx:xx    xxxxxxxx:xx
         xxxxxxxx:xx    xxxxxxxx:xx
         xxxxxxxx:xx    xxxxxxxx:xx
```

```
FIELD DESCRIPTOR
----------------
FLDNAME =xxxxxxxx    ASSOCFIL=x
SUBFILE =x           INVFILE =x
READONLY=x           INFILE  =x
VARFLD  =x           BITFLD  =x
NUMALIGN=x           VARBIT  =x
UNIQUELT=x           INDEXEXT=x
GENERCRT=xxxxxxxx    VALIDRTN=xxxxxxxx
REFORMAT=xxxxxxxxx   FLDPOSIT=xxxxx
FLDEN   =xxxxx       DFLDLEN =xxxxx
FLDLEN  =xxxxx       DELTLIM =xxxxx
ELTLIM  =xxxxx       DELTLEN =xxxxx
SPARE   =xxxxxxxxxxxxxxxx
VALIDARG=xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
         xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
         xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
         xxxxxxx
NAMEFLD =x.xxxxxxxx    x.xxxxxxxx
         x.xxxxxxxx    x.xxxxxxxx
         x.xxxxxxxx    x.xxxxxxxx
         x.xxxxxxxx    x.xxxxxxxx
         x.xxxxxxxx    x.xxxxxxxx
         x.xxxxxxxx    x.xxxxxxxx
         x.xxxxxxxx    x.xxxxxxxx
         x.xxxxxxxx    x.xxxxxxxx
SECURITY=xxxxxxxx    xxxxxxxx    xxxxxxxx
         xxxxxxxx    xxxxxxxx    xxxxxxxx
         xxxxxxxx    xxxxxxxx    xxxxxxxx
         xxxxxxxx    xxxxxxxx    xxxxxxxx
         xxxxxxxx    xxxxxxxx    xxxxxxxx
         xxxxxxxx    xxxxxxxx    xxxxxxxx
```

TOPIC D.18 - MAINTENANCE

A. DATA SET NAME:

DEFIELD which consists of the external structure FIELD

B. CREATED BY:

Not Applicable

C. TYPE OF FILE:

Table

D. ORGANIZATICN:

PL/I Data Structure

E. KEY IDENTIFIER (CCNTROL FIELD):

Not Applicable

F. RECORD LENGTH:

Not Applicable

G. BLOCKING FACTOR:

Not Applicable

H. PURPOSE:

This table is used to contain the names and core locations of all the field descriptor during the running of a retrieval session.

I. PL/I DECLARATION:

THIS STRUCTURE IS USED TO CONTAIN THE FIELD NAMES AND THEIR RESPECTIVE FLD POINTERS.

```
1 FIELD          BASED (X.FIELD_PTR), /* FIELD NAMES AND    */
                              /*       POINTERS STRUCTURE.  */
    3 RECLEN          BIN (31) FIXED, /* RECORD LENGTH FOR  */
                               /* ASMPUT. THIS IS USED IN   */
                               /* CHKPOINT COMMAND. IT IS   */
                               /* SET EQUAL TO ELT LENGTH OF */
                               /*    FIELD STRUCTURE.       */
    3 LAST        BIN FIXED,  /* INDEX OF LAST TABLE ENTRY. */
    3#            BIN FIXED,  /* NUMBER OF ENTRIES IN TABLE.*/
    3 A (X.#FN REFER (FIELD.#)),
        5 NAME CHAR (8),      /* FIELDNAME ARRAY.           */
        5 PTR  PTR;           /* FLD STRUCTURE POINTERS.    */
```

TOPIC D.19 - MAINTENANCE

A. DATA SET NAME:

DERECSEC which consists of the structures RECSEC and RECSEC STR

B. CREATED BY:

Not Applicable

C. TYPE OF FILE:

Table

D. ORGANIZATICN:

PL/I Data Structure

E. KEY IDENTIFIER (CCNTROL FIELD):

Not Applicable

F. RECORD LENGTH:

Not Applicable

G. BLOCKING FACTOR:

Not Applicable

H. PURPOSE:

The RECSEC structure is used to contain the record security codes that pertain to a given file. RECSEC_STR is a character string overlay of the RECSEC structure.

I. PL/I DECLARATION:

THE RECSEC STRUCTURE IS USED TO STORE THE RECORD SECURITY CODES AND SECURITY MASKS USED TO DETERMINE RECORD SECURITY. THE RECSEC STRUCTURE IS POINTED TO BE HDR.RSECTYCD FIELD WHEN THE FILE HAS RECORD SECURITY DEFNED ON IT.

```
1 RECSEC          BASED (X.RSEC_PTR), /* RECORD SECURITY     */
                                      /*    CODES STRUCTURE. */
    3 #           BIN FIXED,          /* NUMBER OF SECURITY CODES. */
    3 SECURITY (18),
        5 CODES CHAR (8),             /* USER PASSWORD.       */
        5 MASK  CHAR (2),             /* RECORD ACCESS CODE.  */
    3 CHANGED (18) BIT (1),           /* ONE FLAG FOR EACH SECURITY */
                                      /* CODE. IF ON THEN REPUT NEW */
                                      /*    VALUE.            */
    3 FILLER      CHAR (8);           /* NEEDED FOR PLI BUG.  */
```

THIS STRUCTURE IS A CHARACTER STRING OVERLAY OF THE RECSEC STRUCTURE. IT IS USED FOR MAKING COPIES OF THE RECSEC STRUCTURE.

```
DCL RECSEC_STR      CHAR (193) BASED (X.RSEC_PTR);
                               /* RECSEC STRUCTURE OVERLAY. */
```

TOPIC D.20 - MAINTENANCE

A. DATA SET NAME:

DESECUR which consists of the structure SECURITY and SECURITY_STR.

B. CREATED BY:

Not Applicable

C. TYPE OF FILE:

Table

D. ORGANIZATION:

PL/I Data Structure

E. KEY IDENTIFIER (CONTROL FIELD):

Not Applicable

F. RECORD LENGTH:

Not Applicable

G. BLOCKING FACTOR:

Not Applicable

H. PURPOSE:

The SECURITY structure is used to contain the security codes defining field security for a given field. SECURITY_STR is a character string overlay of the SECURITY structure.

I. PL/I DECLARATION

THE SECURITY STRUCTURE IS USED TO STORE THE FIELD SECURITY CODES DEFINED FOR A GIVEN FIELD. IT IS POINTED TO BY THE FLD.SECURITY FIELD ON WHICH THIS FIELD SECURITY IS DEFINED.

```
1 SECURITY       BASED (X.FSEC_PTR), /* FIELD SECURITY      */
                                     /*    CODES STRUCTURE.  */
    3 #          BIN FIXED,          /* NUMBER OF SECURITY CODES */
                                     /*    FOR THIS FIELD.   */
    3 CODE (18)  CHAR (8),           /* SECURITY CODE VALUES.  */
    3 CHANGED (18) BIT (1),          /* ONE FLAG FOR EACH SECURITY */
                                     /* CODE. IF ON THEN REPUT THE */
                                     /*    NEW VALUE.        */
    3 FILLER     CHAR (8);           /* NEEDED FOR PLI BUG.    */
```

THIS STRUCTURE IS A CHARACTER STRING OVERLAY OF THE SECURITY STRUCTURE. IT IS USED FOR MAKING COPIES OF THE SECURITY STRUCTURE.

```
DCL SECURITY_STR     CHAR (157) BASED (X.FSEC_PTR);
                                /* SECURITY STRUCTURE OVERLAY.*/
```

TOPIC D.21 - MAINTENANCE

A. DATA SET NAME:

DESUPER which consists of the structure SUPER and SUPER_STR.

B. CREATED BY:

Not Applicable

C. TYPE OF FILE:

Table

D. ORGANIZATION:

PL/I Data Structure.

E. KEY IDENTIFIER (CONTROL FIELD):

Not Applicable

F. RECORD LENGTH:

Not Applicable

G. BLOCKING FACTOR:

Not Applicable

H. PURPOSE:

The SUPER structure is used to contain the superfield component information of a field descriptor. SUPER_STR is a character string overlay of the SUPER structure.

I. PL/I DECLARATION:

THE SUPER STRUCTURE IS USED TO STORE THE SUPERFIELD COMPONENTS OF A SUPER DESCRIPTOR. IT IS POINTED TO BY THE FLD.NAMEFLD OF THE DEFINING SUPERFIELD.

```
1 SUPER          BASED (X.SUPER_PTR), /* SUPER FIELD          */
                                      /* COMPONENT FIELDNAMES  */
                                      /*    STRUCTURE.         */
    3 #          BIN FIXED,  /* NUMBER OF COMPONENT NAMES.    */
    3 NAME (16),
        5 CODE CHAR (1),     /* INTERNAL-EXTERNAL INDICATOR*/
        5 FIELD CHAR(8),     /* COMPONENT FIELD NAMES.        */
    3 CHANGED (16) BIT (1), /* ONE FLAG FOR EACH COMPONENT*/
                            /* NAME. IF ON THEN REPUT THIS*/
                            /*    COMPONENT NAME.            */
    3 FILLER     CHAR (8);  /* NEEDED FOR PLI BUG.           */
```

THIS STRUCTURE IS A CHARACTER STRING OVERLAY OF THE SUPER STRUCTURE. IT IS USED FOR MAKING COPIES OF THE SUPER STRUCTURE.

```
DCL SUPER_STR        CHAR (156) BASED (X.SUPER_PTR);
                        /* SUPER STRUCTURE OVERLAY.   */
```

TOPIC D.22 - MAINTENANCE

A. DATA SET NAME:

DEVALID which consists of the structure VALID

B. CREATED BY:

Not Applicable

C. TYPE OF FILE:

Table

D. ORGANIZATION:

PL/I Data Structure

E. KEY IDENTIFIER (CONTROL FIELD):

Not Applicable

F. RECORD LENGTH:

Not Applicable

G. BLOCKING FACTOR:

Not Applicable

H. PURPOSE:

This structure is used to contain the validation argument for a field descriptor.

I. PL/I DECLARATION:

THE VALID STRUCTURE IS USED TO STORE A VALIDATION ARGUMENT IF ONE IS DEFINED FOR THE FIELD. IT IS POINTED TO BY FLD.VALIDARG IN THE FIELD TO WHICH THIS ARGUMENT BELONGS.

```
1 VALID          BASED (X.ARG_PTR), /* VALIDATION ARGUMENT */
                                    /*    STRUCTURE.       */
    3 LNGTH      BIN FIXED,         /* LENGTH OF VALIDATION */
                                    /*    ARGUMENT.        */
    3 ARGUMENT   CHAR (X.LVA REFER (VALID.LNGTH));
                                    /* VALIDATION ARGUMENT. */
```

TOPIC D.23 - MAINTENANCE

A. DATA SET NAME:

DEFLD which consists of the based structures

FLD and FLD STRING

B. CREATED BY:

Not Applicable

C. TYPE OF FILE:

Table

D. ORGANIZATION:

PL/I Data Structure

E. KEY IDENTIFIER (CONTROL FIELD):

Not Applicable

F. RECORD LENGTH

Not Applicable

G. BLOCKING FACTOR:

Not Applicable

H. PURPOSE:

The FLD structure is used to contain the information describing a field descriptor. FLD_STRING is a character string used to overlay the FLD structure.

I. PL/I DECLARATION:

THIS STRUCUTRE IS USED TO STORE THE INFORMATION DEFINING A FIELD DESCRIPTOR.

```
1 FLD              BASED (X,FLD_PTR), /* FIELD DESCRIPTOR     */
                                      /*    STRUCTURE.        */
    3 BACKWARD     PTR,               /* BACKWARD FLD POINTER. */
    3 FORWARD      PTR,               /* FORWARD FLD POINTER.  */
    3 FLDNAME      CHAR (8),          /* FIELD NAME.           */
    3 ASSOCFIL     CHAR (1),          /* ASSOCIATE FILE IDENTIFIER. */
    3 SUBFILE      CHAR (1),          /* SUBFILE IDENTIFIER.   */
    3 INVFILE      CHAR (1),          /* INVERTED FILE IDENTIFIER. */
    3 READONLY     CHAR (1),          /* FIELD READ ONLY FLAG. */
    3 SUBCNTRL     CHAR (1),          /* FIELD A SUBFILE CONTROL FLD*/
    3 VARFLD       CHAR (1),          /* VARYING LENGTH FIELD FLAG. */
    3 BITFLD       CHAR (1),          /* FIELD IS FIXED LENGTH */
                                      /* BIT STRING OF LENGTH ONE. */
    3 NUMALIGN     CHAR (1),          /* FIELD ALIGNMENT FLAG. */
    3 VARELT       CHAR (1),          /* FIELD ELEMENTS OF VARYING */
                                      /*    LENGTH FLAG.       */
    3 UNIQUELT     CHAR (1),          /* ELEMENTS UNIQUE FLAG. */
    3 INDEXEXT     CHAR (1),          /* INDEX KEYS TO BE IN   */
                                      /* INTERNAL OR EXTERNAL FORM */
                                      /*    FLAG.              */
    3 FILLER       CHAR (1),          /* BOUNDRY ALIGNMENT.    */
    3 GENERCRT     CHAR (8),          /* CONVERSION ROUTINE NAME. */
    3 VALIDRTN     CHAR (8),          /* VALIDATION ROUTINE NAME. */
    3 REFORMAT     CHAR (8),          /* FORMATTING ROUTINE NAME. */
    3 SPARE        CHAR (16),         /* UNUSED DESCRIPTOR FIELD. */
    3 FLDPOSIT     BIN FIXED,         /* FIELD POSITION VALUE. */
    3 FLDLEN       BIN FIXED,         /* FIELD LENGTH VALUE.   */
    3 DFLDLEN      BIN FIXED,         /* MAXIMUM FIELD LENGTH OF ALL*/
                                      /* VALUES STORED ON THE DATA */
                                      /*    BASE.              */
    3 ELTLIM       BIN FIXED,         /* MAX NUMBER OF ELEMENTS/FLD.*/
    3 DELTLIM      BIN FIXED,         /* MAXIMUM ELEMENTS STORED IN */
                                      /* THIS FIELD IN THE DATA BASE*/
    3 ELTLEN       BIN FIXED,         /* ELEMENT LENGTH VALUE. */
    3 DELTLEN      BIN FIXED,         /* MAXIMUM ELEMENT LENGTH */
                                      /* OF ALL OF THE ELEMENTS */
                                      /* STORED FOR THIS  FIELD IN */
                                      /*    THE DATA BASE.     */
    3 VALIDARG     PTR,               /* POINTER TO VALIDATION */
                                      /*    ARGUMENT IF ANY.   */
    3 NAMEFLD      PTR,               /* POINTER TO LIST OF FIELD */
                                      /* NAMES MAKING UP SUPER FIELD*/
    3 SECURITY     PTR,               /* POINTER TO FIELD SECURITY */
                                      /*    CODES IF ANY.      */
    3 BASEFLD      CHAR (8),          /* THE FIELDNAME ON WHICH A */
                                      /* SUBFIELD IS TO BE DEFINED. */
    3 OFFSET       BIN FIXED,         /* THE OFFSET WITHIN THE BASE */
                                      /* FIELD THAT THE SUBFIELD */
```

```
                                /*    STARTS.                    */
      3 FILE_LIST BIN FIXED,    /* ON WHICH ENTRY IN FLD_TAB   */
                                /* HAS THIS FIELD BEEN HUNG.   */
      3 FLDTYPE   BIN FIXED,    /* ENTRY INTO FIELD TYPE TABLE*/
                                /* DEFINING WHICH TYPE OF      */
                                /* FIELD THIS IS.              */
      3 CHANGED (28) BIT (1),   /* ONE FLAG FOR EACH ITEM IN   */
                                /* FLD STRUCTURE. IF ON THEN   */
                                /* PUT NEW VALUE IN DESCRIPTOR*/
                                /*    FILE.                    */
      3 FILLER2         CHAR (8);/* NEEDED FOR PL/I BUG.      */

       THIS STRUCTURE IS A CHARACTER STRING OVERLAY ON THE FLD
       STRUCTURE.  IT IS  USED  FOR  MAKING COPIES  OF THE FLD
       STRUCTURE.

DCL FLD_STRING      CHAR (122) BASED (X.FLD_PTR);
                                /* FLD STRUCTURE OVERLAY.      */
```

TOPIC D.24 - MAINTENANCE

A. DATA SET NAME:

DEXINIT which ccnsists of the X external data structure including all initialization values.

B. CREATED BY:

Not Applicable

C. TYPE OF FILE:

Table

D. ORGANIZATICN:

PL/I Data Structure

E. KEY IDENTIFIER (CCNTROL FIELD):

Not Applicable

F. RECORD LENGTH:

Not Applicable

G. BLOCKING FACTOR:

Not Applicable

H. PURPOSE:

The X structure is used to contain common variables and information used to control the flow through the descriptor editor

TOPIC D.25 - MAINTENANCE

A. DATA SET NAME:

DEX which consists of the X external data structure minus all initialization values.

B. CREATED BY:

Not Applicable

C. TYPE OF FILE:

Table

D. ORGANIZATION:

PL/I Data Structure

E. KEY IDENTIFIER (CONTROL FIELD):

Not Applicable

F. RECORD LENGTH:

Not Applicable

G. BLOCKING FACTOR:

Not Applicable

H. PURPOSE:

The X structure is used to contain common variables and information used to control the flow through the descriptor editor.

I. PL/I DECLARATION:

THE X STRUCTURE IS A COLIECTION OF MINOR STRUCTURES AND SINGLE VARIABLES USED IN THE RUNNING OF THE DESCRIPTOR EDITOR. THESE MINOR STRUCTURES CONSIST OF PREDEFINED FLD, HDR, RECSEC SECURITY, AND SUPER STRUCTURES, AS WELL AS THE INPUT OUTPUT WORK AREAS FOR THE VARIOUS STRUCTURES. THE OTHER MINOR STRUCTURES ARE A LIST OF RESERVED FIELD NAMES AND A LIST OF FIELDS WHICH ARE TO BE DELETED FROM THE DESCRIPTOR FILE WHEN THE CURRENT DESCRIPTORS ARE FILED TO MAKE THE DESCRIPTOR FILE ACCURATE.

```
1 X                EXT CTL, /* EXTERNAL CONTROLLED        */
                            /*   BLOCKED VARIABLES.       */
```

THIS MINOR STRUCTURE IS THE PREDEFINED COMMENTS FIELD DESCRIPTOR.

```
 3 FLD_COMMENTS  LIKE FLD,/*COMMENTS FIELD DESCRIPTOR*/
```

THIS MINOR STRUCTURE IS THE PREDEFINED FREEFORM FIELD DESCRIPTOR.

```
3 FLD_FREEFORM  LIKE FLD, /*USER ENTERED KEYWORDS    */
```

THIS MINOR STRUCTURE IS THE PREDEFINED RECORD SECURITY FIELD DESCRIPTOR.

```
3 FLD_RS        LIKE FLD,
                          /* RECORD SECURITY DESCRIPTOR.*/
```

THIS MINOR STRUCTURE IS THE PREDEFINED BASE FOR A SUBFILE CONTROL FIELD DESCRIPTOR.

```
3 FLD_SUBCNTRL  LIKE FLD,
```

THIS MINOR STRUCTURE IS THE PREDEFINED BASE FOR A SUBFILE KEY FIELD DESCRIPTOR.

```
3 FLD_SUBID     LIKE FLD,
```

THIS MINOR STRUCTURE IS THE PREDEFINED BASE FOR A SUBFILE PARENT KEY FIELD DESCRIPTOR.

```
3 FLD_SUBPK     LIKE FLD,
```

THIS MINOR STRUCTURE IS THE PREDEFINED HEADER DESCRIPTOR RECORD FOR THE ASSOCIATE FILE CONTAINING COMMENTS AND FREEFORM FIELD DESCRIPTORS.

```
3 HDR ASSOC     LIKE HDR,/* HEADER FOR COMMENTS AND  */
```

```
THIS MINOR   STRUCTURE   IS  THE   PREDEFINED   HEADER
DESCRIPTOR RECORD  THE INDEX  FILE ON  WHICH THE  FIELD
FREEFORM IS INDEXED.

3 HDR_INDEX      LIKE HDR,
                        /* INDEX FILE HEADER FOR USER */
                        /* KEYWORDS STORED IN FREEFORM*/

THIS  MINOR STRUCTURE  IS A PREDEFINED INITIALI ED  FLD
STRUCTURE.  IT  IS USED TO INITIALI E A NEWLY ALLOCATED
FLD STRUCTURE.

3 INIT_FLD      LIKE FLD,
                       /* FIELD DESCRIPTOR INITIAL   */
                       /*    VALUES.                 */

THIS MINOR  STRUCTURE IS A  PREDEFINED  INITIALI ED HDR
STRUCTURE.  IT IS USED TO  INITIALI E A NEWLY ALLOCATED
HDR STRUCTURE.

3 INIT_HDR       LIKE HDR,
                        /* HEADER DESCRIPTOR INITIAL */
                        /*    VALUES.                */

THIS  MINOR  STRUCTURE  IS   A  PREDEFINED  INITIALI ED
SECURITY STRUCTURE.  IT IS USED TO INITIALI E  A  NEWLY
ALLOCATED SECURITY STRUCTURE.

3 INIT_SECURITY,        /* FIELD SECURITY STRUCTURE   */
                        /*    INITIAL VALUES.         */
    5 #          BIN FIXED,
    5 CODE (18) CHAR (8)
    5 CHANGED (18) BIT (1),
    5 FILLER    CHAR (8),/* NEEDED FOR PLI BUG.       */

THIS MINOR STRUCTURE  IS USED FOR ALL  IO OPERATIONS TO
AND FROM THE DESCRIPTOR FILE INVOLVING FIELD DESCRIPTOR
RECORDS.    ALL   FIELD   DESCRIPTOR   INPUT   FROM   THE
DESCRIPTOR  FILE  IS PLACED INTO THIS WORK AREA  BEFORE
BEING  MOVED  TO AN  ALLOCATED  FLD  STRUCTURE.  BEFORE
OUTPUTTING TO  A  FIELD  DESCRIPTOR ON  THE  DESCRIPTOR
FILE,  THE FIELD INFORMATION  IS MOVED INTO THE  IO_FLD
STRUCTURE.   THIS  IS NECESSARY BECAUSE  DBPAC REQUIRES
ALL  IO  INTO  AND FROM TO BE  DONE FROM VARYING LENGTH
CHARACTER STRINGS.

3 IO-FLD,               /* FIELD DESCRIPTOR WORK AREA */
                        /*    STRUCTURE.              */
    5 BACKWARD  PTR,    /* BACKWARD FIELD POINTER.    */
    5 FORWARD   PTR,    /* FORWARD FIELD POINTER.     */
    5 FLDNAME   CHAR (8) VAR,/* FIELD NAME.           */
    5 ASSOCFIL  CHAR (1) VAR,/* ASSOCIATE FILE ID.    */
    5 SUBFILE   CHAR (1) VAR,/* SUBFILE IDENTIFIER.   */
```

```
    5 INVFILE    CHAR (1) VAR,/* INVERTED FILE ID.     */
    5 READONLY   CHAR (1) VAR,/* FIELD READ ONLY FLAG.*/
    5 SUBCNTRL   CHAR (1) VAR,/* FIELD IS A SUBFILE   */
                   /*     CONTROL FIELD.             */
    5 VARFLD     CHAR (1) VAR,/* VARGING LENGTH FIELD.*/
    5 BITFLD     CHAR (1) VAR,/* FIXED LENGTH BIT     */
                   /* STRING OF LENGTH ONE.          */
    5 NUMALIGN   CHAR (1) VAR,/* FIELD ALIGNMENT FLAG.*/
    5 VARELT     CHAR (1) VAR,/* FIELD ELEMENTS OF    */
                   /*     LENGTH FLAG.               */
    5 UNIQUELT   CHAR (1) VAR,/* ELEMENTS UNIQUE FLAG.*/
    5 INDEXEXT   CHAR (1) VAR,/* INDEX KEYS TO BE IN  */
                   /* INTERNAL OR EXTERNAL FORM      */
                   /*     FLAG.                      */
    5 FILLER     CHAR (1) VAR,/* BOUNDRY ALIGNMENT.   */
    5 GENERCRT   CHAR (8) VAR,/* CONVERSION RTN NAME. */
    5 VALIDRTN   CHAR (8) VAR,/* VALIDATION RTN NAME. */
    5 REFORMAT   CHAR (8) VAR,/* FORMATTING RTN NAME. */
    5 SPARE CHAR (8) VAR,/* UNUSED DESCRIPTOR FIELD. */
    5 FLDPOSIT   CHAR (2) VAR,/* FIELD POSITION VALUE.*/
    5 FLDLEN     CHAR (2) VAR,/* FIELD LENGTH VALUE.  */
    5 DFLDLEN    CHAR (2) VAR,/* MAXIMUM FIELD LENGTH.*/
    5 ELTLIM     CHAR (2) VAR,/* MAX NUMBER OF        */
                   /*     ELEMENTS / FIELD.          */
    5 DELTLIM    CHAR (2) VAR,/* MAXIMUM # OF ELEMENTS*/
    5 ELTLEN     CHAR (2) var,/* ELEMENT LENGTH VALUE.*/
    5 DELTLEN    CHAR (2) VAR,/* MAXIMUM ELEMENT LNGTH*/
    5 VALIDARG   PTR,   /* POINTER TO VALIDATION     */
                   /*     ARGUMENT IF ANY.           */
    5 NAMEFLD    PTR,   /* POINTER TO LIST OF FIELD  */
                   /* NAMES MAKING UP SUPER FIELD*/
    5 SECURITY   PTR,   /* POINTER TO FIELD SECURITY */
                   /*     CODES IF ANY.              */
    5 BASEFLD    CHAR (8),/* SUBFIELD DEFINING BASE. */
    5 OFFSET     BIN FIXED,/* OFFSET IN BASEFIELD    */
                   /*     SUBFIELD IS TO START.      */
    5 FILE_LIST BIN FIXED,/* WHICH ENTRY IN FLD_TAB. */
    5 FLDTYPE  BIN FIXED,/* TYPE OF FIELD.           */
    5 CHANGED (28) BIT (1),/* ONE FLAG FOR EACH ITEM */
                   /* IN FLD STRUCTURE.  IF ON      */
                   /* THEN PUT NEW VALUE IN         */
                   /*     DESCRIPTOR FILE.           */
    5 FILLER2    CHAR (8),/* NEEDED FOR PLI BUG.     */
```

THIS MINOR STRUCTURE IS USED FOR ALL IO OPERATIONS TO AND FROM THE DESCRIPTOR FILE INVOLVING HEADER RECORDS. ALL INPUT FROM THE DESCRIPTOR FILE INVOLVING HEADER RECORDS IS PLACED IN THE IO_HDR STRUCTURE BEFORE BEING MOVED TO AN ALLOCATED HDR STRUCTURE. TO OUTPUT A HEADER RECORD, THE INFORMATION IS MOVED INTO THE IO-HDR STRUCTURE BEFORE BEING PLACED ON THE FILE. THIS IS NECESSARY BECAUSE DBPAC REQUIRES ALL IO TO BE DONE IN INTO AND FROM VARYING LENGTH CHARACTER STRINGS.

```
3 IO_HDR,                    /* HEADER DESCRIPTOR WORK AREA*/
    5 BACKWARD   PTR,        /* BACKWARD HEADER POINTER.   */
    5 FORWARD    PTR,        /* FORWARD HEADER POINTER.    */
    5 SUFFIX     CHAR (1) VAR,/* WHICH FILE THIS           */
                             /*    HEADER BELONGS TO.      */
    5 FILETYPE   CHAR (1) VAR,/* TYPE OF FILE INDICATR*/
    5 DESCRCT    CHAR (2) VAR,/* NUMBER OF FIELD           */
                             /* DESCRIPTORS ON THIS FILE.  */
    5 BSELNGTH   CHAR (2) VAR,/* TOTAL LENGTH OF FIXED*/
                             /*    FIELDS ON THIS FILE.    */
    5 DESCOK     CHAR (1) VAR,/* DESCRIPTORS OK FLAG. */
    5 SPANNED    CHAR (1) VAR,/* THIS INDEX TO CONSIST*/
                             /* OF SPANNED RECORDS FLAG.   */
    5 DATA       CHAR (1) VAR,/* DATA IS ON FILE FLAG.*/
    5 MNTNABLE   CHAR (1) VAR,/* FILE CAN BE              */
                             /*    MAINTAINED FLAG.        */
    5 MNTNING    CHAR (1) VAR,/* FILE BEING MAINTAINED*/
                             /*    FLAG.                   */
    5 LOADABLE   CHAR (1) VAR,/* FILE CAN BE LOADED.     */
    5 REMAINS    CHAR (4) VAR,/* UNUSED DESCRIPTOR FLD*/
    5 RECSECFP   CHAR (2) VAR,/* FILE HAS RECORD          */
                             /*    SECURITY FLAG.          */
    5 RSECTYCD   PTR,        /*  POINTER TO RECORD        */
                             /*    SECURITY CODES IF ANY.  */
    5 CHANGED (13) BIT (1),/* ONE FLAG FOR EACH ITEM */
                             /* IN HEADER STRUCTURE.  IF   */
                             /* ON THEN PUT NEW VALUE      */
                             /*    IN THE DESCRIPTOR FILE. */
    5 FILLER     CHAR (8),/* NEEDED FOR PLI BUG.          */
```

THIS MINOR STRUCTURE IS USED FOR ALL IO OPERATIONS TO AND FROM THE DESCRIPTOR FILE INVOLVING FIELD SECURITY CODES.

```
3 IO_SECURITY,               /* FIELD SECURITY STRUCTURE.  */
    5 #          BIN FIXED,/* NUMBER OF SECURITY CODES*/
                             /*    FOR THIS FIELD.         */
    5 CODE (18) CHAR (8) VAR,/* USER PASSWORD.           */
    5 CHANGED (18) BIT (1),/* ONE FLAG FOR EACH          */
                             /* SECURITY CODE.  IF ON THEN */
                             /*    REPUT THE NEW VALUE.    */
    5 FILLER     CHAR (8),/* NEEDED FOR PLI BUG.          */
3 GF,                        /* PARAMETERS TO GET FIELD    */
                             /*    SUBROUTINE.             */
    5 ALLOC_NEW BIT (1),/* ALLOCATE AND INITIALI E A */
                             /*    NEW FLD STRUCTURE.      */
    5 FLD_LEN    BIN FIXED,/* MAXIMUM ALLOWABLE LENGTH*/
                             /*    FOR THE FIELDNAME.      */
    5 FLD_MSG    CHAR (8),/* MSGID TO PROMPT FOR THE   */
                             /*    FIELDNAME.              */
    5 FLD#       BIN FIXED,/* FOR AN EXISTING FIELD,   */
                             /* THE ENTRY IN FIELD STRUCTUR*/
                             /*    ELSE 0.                 */
```

```
    5 NEW_FLD    BIT (1),/* ON - GET A BRAND NEW FIELD*/
                         /* OFF - AN EXISTING FIELD.   */
    5 PRMPT_ERR BIT (1),/* VALUE TO SET TC.PROMPT.ERR*/
    5 RESERVED   BIT (1),/* A RETURNED VALUE INDICATIN*/
                         /* IF THE FIELDNAME IS        */
                         /*    RESERVED.               */
    5 RES_FLD    BIT (1),/* ON - RESERVED NAMES ARE   */
                         /*    ACCEPTABLE.             */
                         /* OFF - RESERVED NAMES ARE   */
                         /*    NOT ACCEPTABLE.         */
3 ADD_FLAG       BIT (1),/* IN ADD OR CHANGE COMMAND. */
3 ALPHA      CHAR (26), /* ALL ACCEPTABLE ALPHABETIC  */
                         /*    CHARACTERS.             */
3 ALPHANUMERIC  CHAR (36),
                         /* ALL ACCEPTABLE ALPHA-      */
                         /*    NUMERIC CHARACTERS.     */
3 ARG_PTR        PTR,    /* PTR TO VALIDATION ARGUMENT.*/
3 ASSOC_NAMES    CHAR (9),/* ALL POSSIBLE ASSOCIATE   */
                         /*    FILE ID'S.              */
3 COMND_CALL     BIT  (1),/* A COMMAND CALL OR AN     */
                         /*     INTERNAL CALL.         */
3 COMND_NAME     CHAR (8),/* NAME OF COMMAND CALLED.  */
3 ERR_FLAG       BIT (1),/* ERROR FLAG USED FOR       */
                         /* INTER MODULE COMMUNICATION.*/
3 FIELDNAME      CHAR (8),/* FIELD TO BE PROCESSED.   */
3 FIELDTYPE (0:10) CHAR (2),/* ALL VALID FIELD TYPES.*/
3 FIELD_PTR      PTR,    /* PTR TO FIELD NAME STRUCTURE*/
3 FLD_LAST (60)  PTR,    /* PTRS TO LAST FIELD ENTRY  */
                         /*    IN EACH FIELD STRING.   */
3 FLD_PTR        PTR,    /* PTR TO FIELD DESCRIPTROR.  */
3 FLD_TAB (60)   PTR,    /* PTRS TO FIRST FIELD ENTRY  */
                         /*    IN EACH FIELD STRING.   */
3 FLDTYPE        BIN FIXED,/* FIELD TYPE USED BY ADD. */
3 FSEC_PTR       PTR,    /* PTR TO FIELD SECURITY      */
                         /*    STRUCTURE.              */
3 HDR_PTR        PTR,    /* PTR TO FILE DESCRIPTOR.    */
3 HEAD_TAB (36)  PTR,    /* ONE PTR FOR EACH HEADER.   */
3 HEX_CHARS      CHAR (16),
                         /* ALL ACCEPTABLE HEXADECIMAL */
                         /*    CHARACTERS.             */
3 INDEX_NAMES    CHAR (16),
                         /* LIST OF ALL POSSIBLE INDEX */
                         /*    FILE ID'S.              */
3 IOAREA     CHAR (256) VAR,/* COMMON TERMINAL INPUT  */
                         /*    OUTPUT AREA.            */
3 LOAD_FILE      CHAR (1),/* ID OF FILE TO LOAD FROM. */
3 LOAD_ONE       BIT (1),/* LOAD JUST ONE RECORD.     */
3 LVA            BIN FIXED,/* LENGTH OF VALIDATION    */
                         /*    ARGUMENT.               */
3 PAT_FILE       CHAR (1),/* ID OF FILE BEING WORKED  */
                         /* ON BY REVIEW - PATCH.      */
3 PAT_FIELD      CHAR (8),/* NAME OF FIELD BEING      */
                         /* WORKED ON BY REVIEW-PATCH. */
```

```
3 REV_MODE       BIT (1),/* IN REVIEW OR UPDATE MODE. */
3 RSEC_PTR       PTR,    /* PTR TO RECORD SECURITY     */
                         /*    CODES.                  */
3 SAVE_STRING    CHAR (150) VAR,/* AREA TO BUILD       */
                         /*    COMMAND STRINGS.        */
3 SUBFILE_NAMES CHAR (10),
                         /* LIST OF ALL POSSIBLE       */
                         /*    SUB-FILE ID'S.          */
3 SUFFIX         CHAR (36),
                         /* ALL POSSIBLE FILE          */
                         /*    IDENTIFIER SUFFICIES.   */
3 SUPER_PTR      PTR,    /* PTR TO SUPERFIELD COMPONENT*/
3 TRANS,                 /* TRANSITORY CALL LABELS.    */
    5 CALL       CHAR (8),/* ROUTINE TO BE CALLED      */
    5 RET        CHAR (8),/* ROUTINE TO RETURN TO.     */
```

THIS SUBSEQUENT PART OF THE X STRUCTURE IS SEPARATED FROM THE REST OF THE X ITEMS AS THIS PART OF X IS THE PART THAT MUST BE SAVED WHEN USING THE CHKPOINT COMMAND. THIS PREVIOUS INFORMATION OF X NEED NOT BE SAVED, AS IT IS SETUP PROPERLY WHENEVER X IS ALLOCATED, OR THOSE ITEMS WHOSE VALUES MATTERS NOT BETWEEN COMMAND EXECUTION.

```
3 CHKPOINT_RECLEN BIN (31) FIXED,
                         /* OUTPUT RECORD LENGTH FOR   */
                         /* ASMPUT ROUTINE.  IT IS SET */
                         /* SET TO THE LENGTH OF THE   */
                         /* X STRUCTURE THAT MUST BE   */
                         /* SAVED WHEN CHECKPOINT IS   */
                         /*    EXECUTED.               */
```

THIS MINOR STRUCTURE IS THE PREDEFINED RECLEN FIELD DESCRIPTOR. IT IS PLACED IN IN THIS PART OF X BECAUSE IT MAY HAVE FIELD SECURITY APPLIED TO IT.

```
3 FLD_RECLEN     LIKE FLD,
```

THIS MONOR STRUCTURE IS A LIST OF RESERVED FIELDNAMES. THE USER MAY NOT DEFINE BY USE OF THE ADD, SUPERFLD, CREATSUB, ADDLIKE, AND RENAME COMMANDS A FIELD DESCRIPTOR WITH A FIELDNAME THAT APPEARS IN THIS TABLE.

```
3 RESERVED,              /* LIST OF RESERVED FIELDNAMES*/
    5 LAST_#     BIN FIXED INIT (14),
                         /* INDEX OF LAST ENTRY.       */
    5 FIELDNAME (40) CHAR (8),/*RRESERVED FIELDNAMES */
3 ASSOC_LIST     CHAR (9),/* LIST OF ASSOCIATE FILE    */
                         /* ID'S AVAILABLE FOR         */
                         /*    ASSIGNMENT.             */
3 CREATEF        BIT (1),/* CREATE-UPDATE MODE FLAG.   */
3 DATAPLEX       CHAR (6),/* FILE BEING DEFINED.       */
```

```
3 DELETE_FILES  CHAR (36),/* LIST OF DESCRIPTOR      */
                  /* REGIONS TO BE DELETED FROM */
                  /*    DISC.                   */
3 EXIST_FILES   CHAR (36),/* FILE IDS OF ALL FILES   */
                  /*    EXISTING ON DISC.       */
3 FILE_EXISTS   BIT (1),/* DESCRIPTOR FILE EXISTS.   */
3 INDEX_LIST    CHAR (16),/* LIST OF UNASSIGNED INDEX*/
                  /*   FILE ID'S.               */
3 #FN           BIN FIXED,/* NUMBER OF ENTRIES IN    */
                  /*    FIELD STRUCTURE.        */
3 LOAD_ERROR    BIT (1),
                  /* ERROR IN LOADING DESCRIPTRS*/
3 NEED_FILE     BIT (1),
                  /* USER SHOULD FILE TO SAVE.  */
3 SUBFILE_LIST  CHAR (10),/* LIST OF ALL UNASSIGNED  */
                  /*    SUB_FILE ID'S.          */
```

THIS MINOR STRUCTURE IS USED TO STORE THAT INFORMATION NECESSARY TO THE EXECUTION OF MODULES THAT CAN HAVE PAGEABLE INFORMATION DISPLAYS.

```
3 PAGE_INFO,
    5 RTN       LABEL, /* WHAT ROUTINE TO PAGE.      */
    5 RTN_NAME  CHAR (8),
                  /* THE ROUTINE NAME THE PAGING*/
                  /*    MODULE IS TO CALL.      */
   5 PTR        PTR,   /* ADDRESS OF STRUCTURE BEING */
                  /*    DISPLAYED.              */
   5 DIR        CHAR (1),/* PAGING DIRECTION.       */
   5 FILE_ID    CHAR (1),/* SUFFIX OF FILE BEING    */
                  /*    REVIEWED.               */
   5 FLD_NAME   CHAR (8),
                  /*NAME OF FIELD BEING REVIEWD */
   5 !OTE!      BOM FOXED./* LAST ITEM PUT ON SCREEN¬*/
   5 ITEMS      BIN FIXED,/* # OF ITEMS TO DISPLAY.  */
   5 LIMIT      BIN FIXED,/* ALL ITEMS AFTER LIMIT TO*/
                  /* BE DISPLAYED ONE PER LINE. */
   5 #          BIN FIXED,/* # OF PAGE BEING DISPLAYD*/
   5 LAST       BIN FIXED,/* LAST ENTRY USED.        */
   5 START (100) BIN FIXED,/* ITEM # USED TO START  */
                  /*    EACH PAGE.              */
```

THIS MINOR STRUCTURE IS USED TO STORE THOSE FIELD NAMES AND IDS OF THE DESCRIPTOR REGIONS IN WHICH THEY APPEAR THAT MUST BE DELETED FROM THE DESCRIPTOR FILE THE NEXT TIME A FILE IS DONE.

```
3 DELETE,           /* LIST OF FIELD NAMES TO BE  */
                    /*    DELETED FROM THE DISC.  */
    5 KEY_NAME  CHAR (8),/* ANCHOR KEY NAME IF      */
                    /* ANCHOR KEY NAME HAS BEEN   */
                    /*    CHANGED.                */
    5 #         BIN FIXED,/* NUMBER OF FIELDS        */
```

```
                        /*    TO BE DELETED.          */
5 A (100),
    7 FIELD CHAR (8),/* NAMES OF FIELDS TO BE   */
                     /*   DELETED.               */
    7 IDS   CHAR (4);/* IDS OF FILES ON WHICH   */
                     /*   THE FIELD APPEARS.     */
```

TOPIC D.26 - MAINTENANCE

A. DATA SET NAME:

DEHDR which consists of the structures HDR and HDR_STRING

B. CREATED BY:

Not Applicable

C. TYPE OF FILE:

Table

D. ORGANIZATION:

PL/I Data Structure

E. KEY IDENTIFIER (CONTROL FIELD):

Not Applicable

F. RECORD LENGTH:

Not Applicable

G. BLOCKING FACTOR:

Not Applicable

H. PURPOSE:

The HDR structure is used to contain the information describing a file descriptor. HDR_STRING is a character string used to overlay the HDR structure.

I. PL/I DECLARATION:

THE HDR STRUCTURE IS USED STORE THE INFORMATION DEFINING A FILE DESCRIPTOR.

```
1 HDR              BASED (X.HDR_PTR), /* FILE DESCRIPTOR      */
                                      /*    STRUCTURE.        */
    3 BACKWARD     PTR,               /* BACKWARD HDER POINTER. */
    3 FORWARD      PTR,               /* FORWARD HDER POINTER.  */
    3 SUFFIX       CHAR (1),          /* WHICH FILE THIS HEADER */
                                      /*    BELONGS TO.       */
    3 FILETYPE     CHAR (1)           /* TYPE OF FILE INDICATOR. */
    3 DESCRCT      BIN FIXED,         /* NUMBER OF FIELD DESCRIPTORS*/
                                      /*    ON THIS FILE.     */
    3 BSELNGTH     BIN FIXED,         /* TOTAL LENGTH OF FIXED  */
                                      /*    FIELDS ON THIS FILE. */
    3 DESCOK       CHAR (1),          /* DESCRIPTORS OK FLAG.   */
    3 SPANNED      CHAR (1),          /* THIS INDEX TO CONSIST OF */
                                      /*    SPANNED RECORDS.  */
    3 DATA         CHAR (1),          /* DATA ON FILE SWITCH.   */
    3 MNTNABLE     CHAR (1),          /* FILE CAN BE MAINTAINED FLAG*/
    3 MNTNING      CHAR (1),          /* FILE BEING MAINTAINED FLAG.*/
    3 LOADABLE     CHAR (1),          /* WHETHER OR NOT TO PLACE */
                                      /*    DATA ON THIS FILE. */
    3 REMAINS      CHAR (8),          /* UNUSED HDR DESCRIPTOR FIELD*/
    3 RECSECPP     BIN FIXED,         /* FILE HAS RECORD SECURITY. */
    3 RSECTYCD     PTR,               /* POINTER TO RECORD SECURITY */
                                      /*    CODES IF ANY.     */
    3 CHANGED (13) BIT (1),           /* ONE FLAG FOR EACH ITEM IN */
                                      /* HEADER STRUCTURE. IF ON */
                                      /* THEN PUT NEW VALUE IN */
                                      /*    DESCRIPTOR FILE.  */
    3 FILLER       CHAR   (8);        /* NEEDED FOR PLI BUG.    */
```

THIS STRUCTURE IS A CHARACTER STRING OVERLAY ON THE HDR STRUCTURE. IT IS USED FOR MAKING COPIES OF THE HDR STRUCTURE.

```
DCL HDR_STRING  CHAR (46), BASED (X.HDR_PTR);
                                      /* HDR STRUCTURE OVERLAY.  */
```

TOPIC E.1 - TERMINAL SUPPORT

A. DATA SET NAME:

TSPL/I Diagnostics

B. CREATED BY:

TS Preprocessor Function

C. TYPE OF FILE:

(4) Table

D. ORGANIZATION:

Keyed List

E. KEY IDENTIFIER (CONTROL FIELD):

Each diagnostic comment has an identification key having the form: '---ERROR---nn' where nn is a unique identification number.

F. RECORD LENGTH:

Variable

G. BLOCKING FACTOR:

Not Applicable

H. PURPOSE:

TSPL/I Diagnostic comments are generated into mainline source programs by the TS preprocessor function (see Section IV, Topic E.1 of the DWB). They are tabulated here with additional notes for reference. In Paragraph I, single quotes denote that characters from the TS preprocessor function or its argument are filled into the message to make its meaning clearer.

I. TSPL/I DIAGNOSTIC COMMENTS:

TS001 MISSING ARGUMENT ON TSPL/I REFERENCE.

Severe error - a TS preprocessor function reference has no parenthesized argument.

TS002 TSPL/I ARGUMENT DOES NOT BEGIN WITH A '('.

Severe error - a TS preprocessor function reference

does not begin with double left parentheses. Processing of this TS reference was abandoned because the closing right parenthesis would not be able to be found.

TS003 MISSING DELIMITER IN TSPL/I STATEMENT.

Severe error - the right parenthesis at the end of a TS preprocessor function reference has been encountered unexpectedly.

TS004 STATEMENT HAS A MISSING ';'.

Severe error - the right parenthesis at the end of a TS preprocessor function reference has been encountered unexpectedly.

TS005 STATEMENT FOUND FOLLOWING FINISH.

Severe error - the statement has been ignored because it follows the TS ((FINISH;)) reference.

TS006 STATEMENT CONTAINS EXCESS '(' (s).

Severe error - the statement semicolon has been found, but the parentheses are unbalanced. The statement was ignored.

TS007 STATEMENT KEYWORD UNKNOWN.

Severe error - an unknown word was found as the first word of a TSPL/I statement. The statement was ignored.

TS008 'text' STATEMENT CONTAINS INVALID SYNTAX.

The statement type identified by 'text' was found to contain invalid syntax. The statement was ignored.

TS009 EXTRANEOUS TEXT IGNORED.

This message merely means that part of the statement was ignored.

TS010 IMPROPER OR MULTIPLE ENABLE STATEMENTS.

An improper placement of or multiple use of an enable statement has been encountered. The statement was ignored.

** '---NNNNN---**: TSPL/I ERRORS.'

The finish statement has been processed and NNNNN errors were previously detected.

TOPIC E.2 - TERMINAL SUPPORT

A. DATA SET NAME:

Terminal Control Block

B. CREATED BY:

TS Preprocessor Function

C. TYPE OF FILE:

(4) Table

D. ORGANIZATION:

Linear Structure of Fields

E. KEY IDENTIFIER (CONTROL FIELD):

TC

The terminal control block is an automatic internal data table.

F. RECORD LENGTH:

236 Bytes (Hexidecimal EC)

This is the length of the whole control block including the dope vectors.

G. BLOCKING FACTOR:

Not Applicable

H. PURPOSE:

The TC control block is used for communication between mainline programs and TSSUP. The declaration is generated by the TS preprocessor function. For TSPL/I statements in the mainline, the TS preprocessor function generates statements that post fields in TC, such as a prompt message key. At execution time, TSSUP refers to fields in TC and posts error code fields in TC which may subsequently be referenced in the mainline.

I. PL/I DECLARATION:

```
/*    TERMINAL CONTROL BLOCK (TC) FOR (TS2) TSPL/I     */

DECLARE
 1 TC,                          /*DEFINE THE TC STRUCTURE  */
  2 FUNCTION CHAR(8),           /*TS FUNCTION IDENTIFIER   */
                                /*SET BY TS PREPROCESSOR   */
                                /*'ENTRY '=ENTRY           */
                                /*'READ  '=READ            */
                                /*'WRITE '=WRITE           */
                                /*'FLUSH '=FLUSH           */
                                /*'PUT   '=PUT             */
                                /*'PROMPT-C'=COMMAND PROMPT*/
                                /*'PROMPT-D'=DATA PROMPT   */
                                /*'PROMPT-M'=MESSAGE       */
  2 PAGING_ENTRY CHAR(8),       /*TS PAGING ENTRY POINT    */
                                /*SET BY TS PREPROCESSOR   */
                                /*TO NAME OF THE CURRENT   */
                                /*MODULE'S PAGING ENTRY    */
  2 LINE_SIZE FIXED BIN(15),    /*TS LINE WIDTH (BYTES)    */
                                /*SET BY TSSUP ON ENTRY    */
  2 INPUT,                      /*TS SCREEN INPUT FIELDS   */
   3 ERROR BIT(1),              /*READ ERROR BIT SWITCH    */
                                /*SET BY TSSUP AFTER READ  */
                                /*'0'=NO ERROR '1'=ERROR   */
   3 EXTRA_BITS BIT(7),         /*RESERVED BIT SWITCHES    */
  2 OUTPUT,                     /*TS SCREEN OUTPUT FIELDS  */
   3 SIZE FIXED BIN(15),        /*OUTPUT AREA SIZE (LINES) */
                                /*SET BY TSSUP ON ENTRY    */
   3 INDENT FIXED BIN(15),      /*INDENTATION COLUMN NUMBER*/
                                /*SET BY USER AT ANYTIME   */
   3 WRITTEN FIXED BIN(15),     /*PUT OUTPUT COUNT (BYTES) */
                                /*SET BY TSSUP ON OVERFLOW */
                                /*IF AUTO_WRITE IS SET ON  */
   3 DIRECTION BIT(1),          /*PUT DIRECTION BIT SWITCH */
                                /*SET BY TS PREPROCESSOR   */
                                /*'0'=FORWARD '1'=BACKWARD */
   3 PUT_PARTIAL BIT(1),        /*PUT OUTPUT MODE SWITCH   */
                                /*SET BY USER AT ANYTIME   */
                                /*'0'=PUT FULL RECORD ONLY */
                                /*'1'=PUT PARTIAL RECORD   */
   3 AUTO_WRITE BIT(1),         /*PUT END OF BUFFER SWITCH */
                                /*SET BY USER AT ANYTIME   */
                                /*'0'=RETURN TO USER       */
                                /*'1'=AUTOMATIC WRITE      */
   3 WORD_BREAK BIT(1),         /*PUT LINE SPLIT SWITCH    */
                                /*SET BY USER AT ANYTIME   */
                                /*'0'=TRUNCATE AT LINE END */
                                /*'1'=BREAK AT LAST WORD   */
   3 OVERFLOW BIT(1),           /*PUT OVERFLOW BIT SWITCH  */
```

```
                                /*SET BY TSSUP WHEN PUT     */
                                /*CAUSES BUFFER OVERFLOW    */
                                /*'0'=NO OVERFLOW           */
                                /*'1'=OVERFLOW              */
  3 CONTINUATICN BIT(1),        /*PUT CONTINUATION SWITCH   */
                                /*SET BY USER WHEN HE IS    */
                                /*PUTTING CONTINUED DATA    */
                                /*'0'=NO CONTINUATION       */
                                /*'1'=PUT CONTINUED DATA    */
  3 POSITION BIT(1),            /*PUT POSITIONING SWITCH    */
                                /*SET BY TS PREPROCESSOR    */
                                /*'0'=PUT TO NEXT LINE      */
                                /*'1'=PUT TO TOP OF SCREEN  */
  3 MORE_DATA BIT(1),           /*SCREEN OVERFLOW SWITCH    */
                                /*SET BY THE USER WHEN HE   */
                                /*HAS MORE DATA TO OUTPUT   */
                                /*VIA THE PAGING MECHANISM  */
                                /*'0'=NO MORE DATA REMAINS  */
                                /*'1'=MORE DATA AVAILABLE   */
2 PROMPT,                       /*TS SCREEN PROMPT FIELDS   */
  3 SIZE FIXED BIN(15),         /*PROMPT AREA SIZE (LINES)  */
                                /*SET BY TSSUP ON ENTRY     */
  3 MESSAGE_KEY CHAR(8),        /*KEY OF CURRENT MESSAGE    */
                                /*SET BY TS PREPROCESSOR    */
  3 KEYWORD CHAR(8),            /*KEYWORD FOR DATA PROMPT   */
                                /*SET BY TS PREPROCESSOR    */
  3 BYPASS BIT(1),              /*PROMPTING BYPASS SWITCH   */
                                /*SET BY USER AT ANYTIME    */
                                /*'0'=PROMPT IF NO DATA     */
                                /*'1'=RETURN NULL VALUE     */
  3 ERROR BIT(1),               /*PROMPTING ERROR SWITCHZZZ*/
                                /*SET BY USER WHEN A DATA   */
                                /*ERROR FORCES REPROMPTING  */
                                /*'0'=PROCESS NORMALLY      */
                                /*'1'=REPROMPT FOR DATA     */
  3 TRUNCATION BIT(1),          /*DATA TRUNCATION SWITCH    */
                                /*SET BY TSSUP FOR PROMPT   */
                                /*'0'=NO TRUNCATION         */
                                /*'1'=DATA TRUNCATED        */
  3 DEFAULT BIT(1),             /*DEFAULT VALUE BIT SWITCH  */
                                /*SET BY TSSUP FOR PROMPT   */
                                /*'0'=DATA ENTERED BY USER  */
                                /*'1'=DATA WAS A DEFAULT    */
  3  UOTED BIT(1),              /* UOTED  UOTED BIT SWITCH  */
                                /*SET BY TSSUP FOR PROMPT   */
                                /*'0'=NORMAL DATA VALUE     */
                                /*'1'= UCTED STRING         */
  3 MORE_DATA BIT(1),           /*PARENTHIZED LIST SWITCH   */
                                /*SET BY TSSUP FOR PROMPT   */
                                /*'0'=LAST DATA VALUE       */
                                /*'1'=MORE VALUES FOLLOW    */
  3 SKIP BIT(1),                /*SKIP INPUT PARSING BIT    */
                                /*SET BY THE USER WHEN HE   */
```

```
                                /*WISHES TO BYPASS PARSING */
                                /*'0'=DO NORMAL PARSING    */
                                /*'1'=SKIP NORMAL PARSING  */
3 PAGE BIT(1);                  /*PAGING CONTROL SWITCH    */
                                /*'0'=IGNORE PAGING ENTRY  */
                                /*'1'=ALTER PAGING ENTRY   */
```

TOPIC E.3 - TERMINAL SUPPORT

A. DATA SET NAME:

TSTEXT - Terminal Control Block Declaration

B. CREATED BY:

Included by TS processor function.

C. TYPE OF FILE:

Table

D. ORGANIZATION:

PL/I Source Statements

E. KEY IDENTIFIER (CONTROL FIELD):

Not Applicable

F. RECORD LENGTH:

92 bytes

G. BLOCKING FACTOR:

Not Applicable

H. PURPOSE:

The purpose of the TSTEXT is to define the terminal control block (TC) within every program using it. This enables the programmer and the preprocessor to refer to the fields of the TC block in order to specify the various functions and options needed by the program.

I. PL/I DECLARATION:

```
DECLARE
   TSFLUSH ENTRY(),              /* FLUSH ENTRY POINT          */
   TSREAD ENTRY(,CHAR(*) VAR),/* READ ENTRY POINT           */
   TSWRITE ENTRY(,CHAR(*) VAR),/* WRITE ENTRY POINT         */
   TSPUT ENTRY(,CHAR(*) VAR, /* PUT ENTRY POINT            */
               CHAR(*) VAR),/*   DEFINITION               */
   ( TSPRMTC,                /* COMMAND PROMPT ENTRY       */
     TSPRMTD,                /* DATA PROMPT ENTRY          */
     TSPRMTM ) ENTRY(,CHAR(*) VAR, /* MESSAGE ENTRY POINT*/
 CHAR(*) VAR, CHAR(*) VAR, CHAR(*) VAR, CHAR(*) VAR,
 CHAR(*) VAR, CHAR(*) VAR, CHAR(*) VAR, CHAR(*) VAR,
 CHAR(*) VAR, CHAR(*) VAR, CHAR(*) VAR, CHAR(*) VAR,
```

```
CHAR(*) VAR, CHAR(*) VAR, CHAR(*) VAR, CHAR(*) VAR,
CHAR(*) VAR, CHAR(*) VAR, CHAR(*) VAR, CHAR(*) VAR);
   DECLARE
       1 TC,                      /* DEFINE THE TC STRUCTURE  */
2 FUNCTION CHAR(8),               /* TS FUNCTION IDENTIFIER   */
                                  /* SET BY TS PREPROCESSOR   */
                                  /* 'ENTRY    '=ENTRY        */
                                  /* 'READ     '=READ         */
                                  /* 'WRITE    '=WRITE        */
                                  /* 'FLUSH    '=FLUSH        */
                                  /* 'PUT      '=PUT          */
                                  /* 'PROMPT-C'=COMMAND PROMPT*/
                                  /* 'PROMPT-D'=DATA PROMPT   */
                                  /* 'PROMPT-M'=MESSAGE       */
2 PAGING_ENTRY CHAR(8),           /* TS PAGING ENTRY POINT    */
                                  /* SET BY TS PREPROCESSOR   */
                                  /* TO NAME OF THE CURRENT   */
                                  /* MODULE'S PAGING ENTRY    */
2 LINE_SIZE FIXED BIN(15),        /* TS LINE WIDTH (BYTES)    */
                                  /* SET BY TSSUP ON ENTRY    */
2 INPUT,                          /* TS SCREEN INPUT FIELDS   */
    3 ERROR BIT(1),               /* READ ERROR BIT SWITCH    */
                                  /* SET BY TSSUP AFTER READ  */
                                  /* '0'=NO ERROR '1'=ERROR   */
    3 EXTRA_BITS BIT(7),          /* RESERVED BIT SWITCHES    */
2 OUTPUT,                         /* TS SCREEN OUTPUT FIELDS  */
    3 SIZE FIXED BIN(15),         /* OUTPUT AREA SIZE (LINES) */
                                  /* SET BY TSSUP ON ENTRY    */
    3 INDENT FIXED BIN(15),       /* INDENTATION COLUMN NUMBER*/
                                  /* SET BY USER AT ANYTIME   */
    3 WRITTEN FIXED BIN(15),      /* PUT OUTPUT COUNT (BYTES) */
                                  /* SET BY TSSUP ON OVERFLOW */
                                  /* IF AUTO_WRITE IS SET ON  */
    3 DIRECTION BIT(1),           /* PUT DIRECTION BIT SWITCH */
                                  /* SET BY TS PREPROCESSOR   */
                                  /* '0'=FORWARD '1'=BACKWARD */
    3 PUT_PARTIAL BIT(1),         /* PUT OUTPUT MODE SWITCH   */
                                  /* SET BY USER AT ANYTIME   */
                                  /* '0'=PUT FULL RECORD ONLY */
                                  /* '1'=PUT PARTIAL RECORD   */
    3 AUTO_WRITE BIT(1),          /* PUT END OF BUFFER SWITCH */
                                  /* SET BY USER AT ANYTIME   */
                                  /* '0'=RETURN TO USER       */
                                  /* '1'=AUTOMATIC WRITE      */
    3 WORD_BREAK BIT(1),          /* PUT LINE SPLIT SWITCH    */
                                  /* SET BY USER AT ANYTIME   */
                                  /* '0'=TRUNCATE AT LINE END */
                                  /* '1'=BREAK AT LAST WORD   */
    3 OVERFLOW BIT(1),            /* PUT OVERFLOW BIT SWITCH  */
                                  /* SET BY TSSUP WHEN PUT    */
                                  /* CAUSES BUFFER OVERFLOW   */
                                  /* '0'=NO OVERFLOW          */
                                  /* '1'=OVERFLOW             */
```

```
      3 CONTINUATION BIT(1),   /* PUT CONTINUATION SWITCH  */
                               /* SET BY USER WHEN HE IS   */
                               /* PUTTING CONTINUED DATA   */
                               /* '0'=NO CONTINUATION      */
                               /* '1'=PUT CONTINUED DATA   */
      3 POSITION BIT(1),       /* PUT POSITIONING SWITCH   */
                               /* SET BY TS PREPROCESSOR   */
                               /* '0'=PUT TO NEXT LINE     */
                               /* '1'=PUT TO TOP OF SCREEN */
      3 MORE_DATA BIT(1),      /* SCREEN OVERFLOW SWITCH   */
                               /* SET BY THE USER WHEN HE  */
                               /* HAS MORE DATA TO OUTPUT  */
                               /* VIA THE PAGING MECHANISM */
                               /* '0'=NO MORE DATA REMAINS */
                               /* '1'=MORE DATA AVAILABLE  */
    2 PROMPT,                  /* TS SCREEN PROMPT FIELDS  */
      3 SIZE FIXED BIN(15),    /* PROMPT AREA SIZE (LINES) */
                               /* SET BY TSSUP ON ENTRY    */
      3 MESSAGE_KEY CHAR(8),   /* KEY OF CURRENT MESSAGE   */
                               /* SET BY TS PREPROCESSOR   */
      3 KEYWORD CHAR(8),       /* KEYWORD FOR DATA PROMPT  */
                               /* SET BY TS PREPROCESSOR   */
      3 BYPASS BIT(1),         /* PROMPTING BYPASS SWITCH  */
                               /* SET BY USER AT ANYTIME   */
                               /* '0'=PROMPT IF NO DATA    */
                               /* '1'=RETURN NULL VALUE    */
      3 ERROR BIT(1),          /* PROMPTING ERROR SWITCH   */
                               /* SET BY USER WHEN A DATA  */
                               /* ERROR FORCES REPROMPTING */
                               /* '0'=PROCESS NORMALLY     */
                               /* '1'=REPROMPT FOR DATA    */
      3 TRUNCATION BIT(1),     /* DATA TRUNCATION SWITCH   */
                               /* SET BY TSSUP FOR PROMPT  */
                               /* '0'=NO TRUNCATION        */
                               /* '1'=DATA TRUNCATED       */
      3 DEFAULT BIT(1),        /* DEFAULT VALUE BIT SWITCH */
                               /* SET BY TSSUP FOR PROMPT  */
                               /* '0'=DATA ENTERED BY USER */
                               /* '1'=DATA WAS A DEFAULT   */
      3 QUOTED BIT(1),         /* QUOTED QUOTED BIT SWITCH */
                               /* SET BY TSSUP FOR PROMPT  */
                               /* '0'=NORMAL DATA VALUE    */
                               /* '1'=QUOTED STRING        */
      3 MORE_DATA BIT(1),      /* PARENTHIZED LIST SWITCH  */
                               /* SET BY TSSUP FOR PROMPT  */
                               /* '0'=LAST DATA VALUE      */
                               /* '1'=MORE VALUES FOLLOW   */
      3 SKIP BIT(1),           /* SKIP INPUT PARSING BIT   */
                               /* SET BY THE USER WHEN HE  */
                               /* WISHES TO BYPASS PARSING */
                               /* '0'=DO NORMAL PARSING    */
                               /* '1'=SKIP NORMAL PARSING  */
      3 PAGE BIT(1);           /* PAGING CONTROL SWITCH    */
```

```
/* '0'=IGNORE PAGING ENTRY  */
/* '1'=ALTER PAGING ENTRY   */
```

TOPIC F.1 - DATA RETRIEVAL

A. DATA SET NAME

RETDATA - Retrieval Data Table

B. CREATED BY:

RDBINIT

C. TYPE OF FILE:

Table

D. ORGANIZATION

PL/I Data Structure

E. KEY IDENTIFIER (CONTROL FIELD):

Not Applicable

F. RECORD LENGTH:

Not Applicable

G. BLOCKING FACTOR:

Not Applicable

H. PURPOSE

1. RETDATA EXTERNAL CONTROLLED.
This table contains data fields unique to the retrieval sub-system and referenced by various modules of that sub-system.

2. PRINTDD CHARACTER (8) VARYING.
This field contains the ddname of the print file for retrieval.

2. PRINTDS CHARACTER (35) VARYING.
This field contains the dsname of the print file for retrieval.

2. SRT98DD CHARACTER (8) VARYING.
This field contains the ddname of the save file for retrieval.

2. SRT98DS CHARACTER (35) VARYING.
This field contains the dsname of the save file for retrieval.

2. BITS.
The following bit switches are used to communicate status between the various retrieval modules.

3. PRTUSED BIT (1).
Describes whether any data has been written to the retrieval print file.

I. PL/I DECLARATION

```
/*              NASIS SYSTEM RETRIEVAL DATA TABLE          */

DCL
1 RETDATA EXTERNAL CONTROLLED,/*DEFINE RETRIEVAL DATA      */
  2 PRINTDD CHAR(8) VAR,      /*PRINT FILE DDNAME          */
  2 PRINTDS CHAR(35) VAR,     /*PRINT FILE DSNAME          */
  2 SET98DD CHAR(8) VAR,      /*SAVE FILE DDNAME           */
  2 SET98DD CHAR(8) VAR,      /*SAVE FILE DDNAME           */
  2 SET98DS CHAR(35) VAR,     /*SAVE FILE DSNAME           */
  2 BITS,                     /*RETRIEVAL BIT SWITCHES     */
    3 PRTUSED BIT(1),         /*PRINT FILE USED BIT        */
    3 UNUSED BIT(7);          /*UNASSIGNED BIT SWITCHES    */
```

TOPIC F.2 - DATA RETRIEVAL

A. DATA SET NAME:

EXPAND Display Format

B. CREATED BY:

EXPAND (RDEXPND)

C. TYPE OF FILE:

(3) Terminal Communication

D. ORGANIZATION:

Character Display Screen

E. KEY IDENTIFIER (CONTROL FIELD):

Not Applicable

F. RECORD LENGTH:

Variable (Enter output area of the screen or pseudo screen)

G. BLOCKING FACTOR:

Not Applicable

H. PURPOSE:

This is a series of on-line output displays produced by the EXPAND command giving the user full access to the inverted indexes of a data base assisting him in an on-line search for information.

The display is adapted to the size of the display screen being used. If the end of the inverted index or the end of the range of E-numbers (000-999) is encountered in either direction, a line such as

(--START OF INDEX--)
(--END OF INDEX----)

is displayed in the appropriate row. The primary term is always regenerated on the appropriate row when multiple paging operations are done in either direction even if the primary term is not found in the inverted index.

SAMPLE EXPAND DISPLAY

```
SYSTEM: -ENTER:
USER:  expand pli,language
SYSTEM:  LINE XREFS  LANGUAGE(S)
          ****       (--START OF INDEX--)
          E097   28  ASM
          E098    6  ENG
          E099   12  N/A
         -E100   43  PLI
          E101    4  TSS
          ****       (---END OF INDEX---)
```

TOPIC F.3 - DATA RETRIEVAL

A. DATA SET NAME:

SELECT Display Format

B. CREATED BY:

SELECT (RDBSLCT and RDBSETS)

C. TYPE OF FIIE:

(3) Terminal communication

D. ORGANIZATICN:

Character Display Screen

E. KEY IDENTIFIER (CCNTBCL FIELD):

Not Applicable

F. RECORD LENGTH:

480 Byte typical - 40 column, 22 line output area apart from the prompting area.

G. BLOCKING FACTOR:

Not Applicable

H. PURPOSE:

This is the output generated by the SELECT command. RDBSLCT calls the DBPSET entry point of RDBSETS to post the users new set and RDBSETS sends this display to the prompt area of the screen.

SELECT COMMAND SCREEN DISPLAY

```
aa  bbbbb  ccccccc

 or

aa  bbbbb  (FROM: dddddd) ccccccc
```

where:

aa	=	set number
bbbbb	=	number of references
cc etc.	=	SELECT expression
dddddd	=	control field name, if applicable

TOPIC F.4 - DATA RETRIEVAL

A. DATA SET NAME:

DISPLAY Display Format

B. CREATED BY:

DISPLAY (RDBDSPI)

C. TYPE OF FILE:

(3) Terminal Communication

D. ORGANIZATION:

Character Display Screen

E. KEY IDENTIFIER (CONTROL FIELD):

Not Applicable

F. RECORD LENGTH:

480 Bytes typical - 40 column, 12 line output area apart from the prompting area.

G. BLOCKING FACTOR:

Not Applicable

H. PURPOSE:

This is a series of on-line output displays produced by the DISPLAY command giving the user full access to the anchor and associated files of a data base assisting him in an on-line search for information. Each screen image is built in a PAGTAB buffer and then transmitted in a single output operation to the display screen. A special use of the DISPLAY command is to retrieve saved screen images and redisplay them. Usually a stored screen image is one of the formats produced by the various commands, but it may even be a screen image the user has keyed in.

The display is adapted to the size of the display screen being used including the degenerate case of a typewriter terminal (120 columns by one line).

The first row under the heading rows always has a field name tag, even when it is a continuation of an element value begun on the previous screen.

DISPLAY Command Screen Display

```
DISPLAY  aa,b,ccccc  (original command parameters)

 wwwwwwww: mmmmmm CF SET aa, FORMAT b, ITEM nn
 xxxxxxxx: n
 yyyyyyyy: p,
         : q
 zzzzzzzz: rrrrrrrrrrrrrrrrrrrrrrrrrrrrrr
 rrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrr
 rrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrr
 rrrrrrrrrrrrrrrr etc.
```

where:

```
 nn        = relative record in set/key aa
 mmm etc.  = up to 30 characters of key value.
 p, q      = up to 30 characters of element value.
 rrr etc.  = 77 character element value.
 www etc.  = key field name.
 xxx etc.  = field name having a single short element.
 yyy etc.  = field name having two short elements.
 zzz etc.  = field name having a single long element.
```

COLUMNAR

```
DISPLAY aa, FFF, ccccc (original command parameters)
                                         PAGE xx
    _________________________ t1 ______________________
    _________________________ t2 ______________________
                              :
                              :
    _________________________ tn ______________________
    _________________________ h1 ______________________
    _________________________ h2 ______________________
                              :
                              :
    _________________________ hn ______________________

    F1  F2          F5
        F2
            F3  F4
                F4
            F3  F4
            F3
```

where:

```
xx     = page number.
t1     = one or more title lines.
h1     = one or more header lines.
F1,F5  = one element field on the anchor or associate
         file.
F2     = a multi-element field.
F3,F4  = an elemental field on a subfile.
```

TOPIC F.5 - DATA RETRIEVAL

A. DATA SET NAME:

PARSED Table

B. CREATED BY:

Not Applicable

C. TYPE OF FILE:

Table

D. ORGANIZATION:

Linear Structure of elements

E. KEY IDENTIFIER (CONTROL FIELD):

Not Applicable

F. RECORD LENGTH:

Not Applicable

G. BLOCKING FACTOR:

Not Applicable

H. PURPOSE:

PARSED is a group of structures used by SELECT to save the information from a parsed expression, when that expression requires a search or contains "S" numbers which cannot be resolved with set numbers until after execution at the actual linear search.

At search time EXSEARCH calls SELECT to begin evaluation of the boolean expressions and to post sets to be searched. SELECT uses the information contained in PARSED to do this and to replace all "S" numbers with their corresponding set numbers. After the search SELECT proceeds with the final evaluation of the expression.

1. PARSED is a based structure consisting of pointers to the other structures containing the various pieces of information that needs to be saved when a boolean expression contains "S"numbers.

2. PARS_TAB_PTR is a pointer to the structure

which is used to describe each element of the expression.

2. PTAB_INFO_PTR is a pointer to the structure which holds additional information about each element of the expression.

2. PTAB_PTRS_PTR is a pointer to the array of pointers, each pointer corresponding to an element of the expression.

2. INST_LIST_PTR is a pointer to the list of instructions generated by SELECT to provide for evaluation of the expression.

2. WAS_PTR is a pointer to the work string in which the expression and other necessary character strings are stored.

2. INTH is the length for allocation of all tables listed here except WAS. It is determined from the length of the input expression.

2. S# contains the S# in which the PARSED pointer is stored, i.e. the PARSED pointer is stored in SRCHTAB.ENTRYDEF.(PARSED.S#)

1. PARS_TAB is the primary table for storage of information as the expression is parsed.

2. LNTH is the number of array elements in the table.

2. EL is an element of the table. One element in the table is used to describe each syntactic item in the expression.

3. IDX is the relative position of the item in the string WAS.

3. LTH is the length of the item.

3. ID is the identifier of the item which distinguishes between items of the same general type.

3. TYPE is the general type of the item, such a relation operator, character string, etc.

3. TERM is used to mark an item as being a term during expression evaluation, or

to mark an item so that it will be ignored by later passes.

3. SKIP causes later program passes to skip over a particular number of items (or elements in PARS_TAB).

1. PTAB_PTRS is an array of pointers. Each pointer corresponds directly to an element in PARS_TAB_the Nth pointer in PTAB_PTRS corresponds to the Nth element in PARS_TAB. When an expression item results in the formation of a set, the pointer to the set is stored in the corresponding element of PTAB_PTRS.

2. INTH is number of array elements.

2. EL is a pointer array element.

1. PTAB_INFO is a table for storage of additional information about an expression item, and again each element corresponds directly to an element in PARS_TAB.

2. LNTH is number of array elements.

2. EL is an array element.

3. IDX relative position of item, in string WAS, which is associated with item to which this element corresponds.

3. SFX indicates subfile which applies to item.

3. INDXD on if item (Fieldname or value) is indexed.

3. NNDXE on if item (Fieldname or value) is not indexed.

3. CTL on when item (Fieldname) is control field name.

1. INST_LIST is a list of "instructions" created and executed by SELECT. The instructions guide the creation of sets, both from index files and through linear search, as well as the boolean combinatien of all sets, once formed, to yield the final set.

2. LNTH number of instruction elements in this list.

2. EL an instruction.

3. OP is the operation code.

3. IDX1 first parameter/

3. IDX2 second parameter.

3.IDX3 third parameter.

1. WAS is a work string containing the input expression and other necessary character strings.

2. INTH length of work string.

2. S actual string.

1. WAA is a one-character-per-element array which is defined on top of WAS to allow easy access to a single character.

2. LNTH is number of elements.

2. A is a one character element.

PARSED_LIST is base pointer for PARSED structure.

PARS_TAB_PTR is base pointer for PARS_TAB.

PTAB_INFO_PTR is base pointer for PTAB_INFO.

PTAB_PTRS_PTR is base pointer for PTAB_PTRS.

WAS_PTR is base pointer for WAS and WAA.

INST_LIST_PTR is base pointer for INST_LIST.

WAS_SIZE is set to adjust size of WAS at allocation.

TOPIC F.6 - DATA RETRIEVAL

A. DATA SET NAME:

SETS Display Format

B. CREATED BY:

SETS (RDBSETS)

C. TYPE OF FILE:

Terminal Communication

D. ORGANIZATION:

Character Screen Display

E. KEY IDENTIFIER (CONTROL FIELD):

Not Applicable

F. RECORD LENGTH:

320 Bytes typical - 40 column, 8 line output area apart from the prompting area.

G. BLOCKING FACTOR:

Not Applicable

H. PURPOSE:

This is the output created by the SETS command. It is a display on the user's screen or typewriter terminal of the sets created during the current strategy session.

This display consists of the set number or S-number, the number of index references in the set, and the expression (including the control field name, if applicable) that formed the set. The expression will wrap around if it exceeds one line.

Paging forwards and backwards is available. The word 'MORE:' will appear at the bottom of the list if there is more data forward.

I. SAMPLE OUTPUT:

```
ENTER : SETS

SET#       XREFS     EXPRESSION       PAGE 1
 aa        bbbb      ccccccccccccccccccccc
             "       cccccc
             "
             "
             "
             "
 aa        bbbb      (FROM: dddddd) cccccc
-MORE:
```

Where:

```
 aa         = set number,
 bbbb       = number of references,
 ccc etc.   = expression,
 dddddd     = control field name,
-MORE:      = forward continuation indicator.
```

TOPIC F.7 - DATA RETRIEVAL

A. DATA SET NAME:

EXECUTE Display Format

B. CREATED BY:

EXECUTE (RDBEXSR)

C. TYPE OF FILE:

Terminal Communication

D. ORGANIZATION:

Character Display Screen

E. KEY IDENTIFIER (CONTROL FIELD):

Not Applicable

F. RECORD LENGTH:

880 Bytes - typical 40 column, 22 line output area apart from the prompting area.

G. BLOCKING FACTOR:

Not Applicable

H. PURPOSE:

This is the output generated by the EXECUTE command which may appear in the output area of the screen. If set numbers are not being displayed in the output area, then the output from EXECUTE will appear in the prompt area of the screen.

When the EXECUTE output is being placed in the output area of the screen, the screen image is transmitted in a single output operation to the display screen.

The display is adapted to the size of the display screen being used, including the degenerate case of a typewriter terminal (one line).

The output screen may contain from one line to the whole output area as output.

EXECUTE Command Screen Display:

```
aa bbbbb cccccc ...
```

where:

aa = set number
bbbbb = number of references
cccccc ... = SELECT (search option) expression.

TOPIC F.8 - DATA RETRIEVAL

A. DATA SET NAME:

PRINT Data Set Format

B. CREATED BY:

PRINT (RDBPRNT)

C. TYPE OF FILE:

(5) Non-data base file and

(2) Formatted print-out

D. ORGANIZATION:

VSAM

E. KEY IDENTIFIER (CONTROL FIELD):

Not Applicable

F. PRINT LENGTH:

132 Bytes maximum printed plus record length and carriage control fields (5 bytes).

G. BLOCKING FACTOR:

Block size = 4096 bytes.

H. PURPOSE:

This is an output data set produced by the PRINT command. It consists of line images written using a PL/I file named PRINTER. At the end of a terminal session a ISS PRINT task is initiated to print the data set off-line on a line printer.

A leader page shows the user's name and mail stop for distribution. Following the output produced for each PRINT command is a separator page having 36 dollar signs on the first line.

PRINT Command - LEADER PAGE

DISTRIBUTE TO: xxxxxxxxx etc.

MAIL STOP: yyyyyyy etc.

where:

xxx etc. = user's name
yyy etc. = mail stop

PRINT Command - TYPICAL FORMAT 1 PAGE

```
PRINT OF SET xx, Format 1.

aaaaaaaa: ddddddd
aaaaaaaa: eeeeeee
aaaaaaaa: fffffff
```

where:

aaaaaaaa = key field name
d thru f = key value (wraps around to column 1 if more than 122 characters).

PRINT Command - TYPICAL FORMAT 2, 3 or 4 PAGE

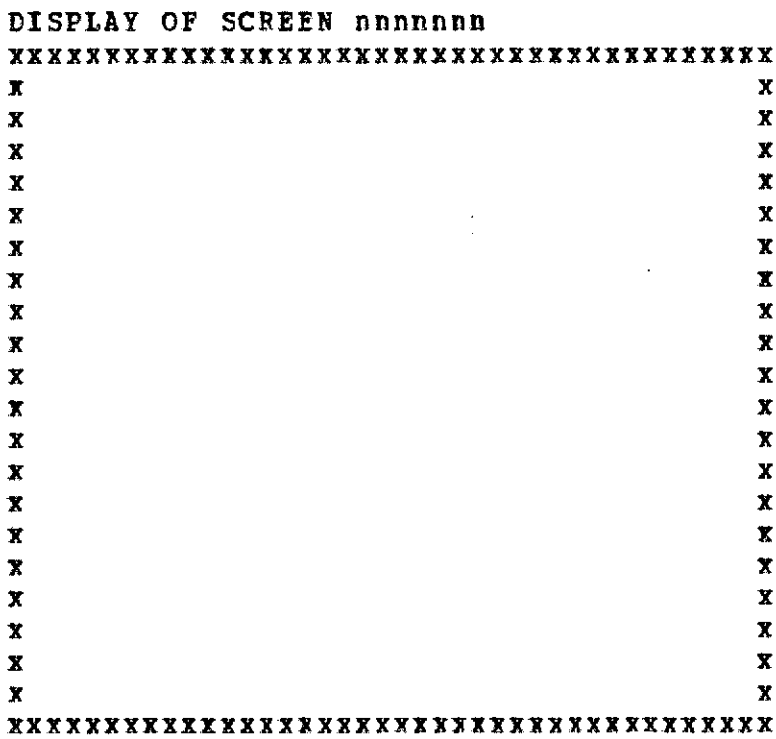
```
PRINT OF SET xx, FORMAT y, zzzzzzzz:vvvvvvvvvvvv PAGE wwwww

aaaaaaaa: d
bbbbbbbb: e
        : f
cccccccc: gggg ..... gggg
ggggggggggggggg
gggggg
```

where:

```
aaaaaaaa = field name having a single short element.
bbbbbbbb = field name having two short elements.
cccccccc = field name having a long element.
d, f     = element value up to 122 characters (no maximum
           number of elements).
ggg etc. = 379 character element value (no maximum).
zzz etc. = key field name.
vvv etc. = first 74 characters of key value.
```

PRINT Command - TYPICAL SET 98 (saved screen image) PAGE



where:

nnnnnn = relative saved screen number.

TOPIC F.9 - DATA RETRIEVAL

A. DATA SET NAME:

EXPTAB - Expand Term Table

B. CREATED BY:

RDBXPND

C. TYPE OF FILE:

(4) Table

D. ORGANIZATION:

PL/I Data Structure

E. KEY IDENTIFIER (CONTROL FIELD):

Not Applicable

F. RECORD LENGTH:

Not Applicable

G. BLOCKING FACTOR:

Not Applicable

H. PURPOSE:

1. EXPTAB EXTERNAL CONTROLLED.
This table contains a list of alphabetically sequential terms taken from an inverted index file and information relating the terms to reference numbers (E-numbers) used in the SELECT command.

2. TERMS AREA (2500).
This area contains a linked list of terms read from the inverted index.

2. FIRST-PRT POINTER.
Points to the first term in the linked list.

2. LAST-PRT POINTER.
Points to the last term in the linked list.

2. TOP-PTR POINTER
Points to the first term displayed on the current page of data.

2. BOTTOM-PTR POINTER
Points to the last term displayed on the current page of data.

2. FIRST - E# BINARY FIXED.
Contains the reference number for the first terms in the list.

2. LAST E# BINARY FIXED.
Contains the reference number for the last term in the list.

2. TOP-E# BINARY FIXED.
Contains the reference number for the first term on the current page.

2. BOTTOM-E# BINARY FIXED.
Contains the reference number for the last term on the current page.

2. LO-E# BINARY FIXED.
Contains the lowest valid reference number. (Either 1 or the reference number of the index origin.)

2. HI-E# BINARY FIXED.
Contains the highest valid reference number. (Either 999 or the reference number of the index end.)

I. PL/I DECLARATION

```
DCL 1 EXPTAB EXT CONTROLLED,/*DEFINE THE TERM TABLE    */
      2 TERMS AREA(2500),   /*TERM STORAGE AREA        */
      2 FIRST_PTR POINTER,  /*FIRST LIST ENTRY POINTER*/
      2 LAST_PTR POINTER,   /*LAST LIST ENTRY POINTER */
      2 TOP_PTR POINTER,    /*FIRST LINE ON PAGE PTR   */
      2 BOTTOM_PTR POINTER, /*LAST LINE ON PAGE PTR    */
      2 FIRST_E# BIN FIXED, /*FIRST ENTRY'S E#         */
      2 LAST_E# BIN FIXED,  /*LAST ENTRY'S E#          */
      2 TOP_E# BIN FIXED,   /*FIRST E# ON PAGE         */
      2 BOTTOM_E# BIN FIXED,/*LAST E# ON PAGE          */
      2 LO_E# BIN FIXED,    /*LOWEST VALID E#          */
      2 HI_E# BIN FIXED;    /*HIGHEST VALID E#         */
```

TOPIC F.10 - DATA RETRIEVAL

A. DATA SET NAME:

FLDTAB - Field Name Table

B. CREATED BY:

DBPFLDT entry of module RDBPAC

C. TYPE OF FILE:

(4) Table

D. ORGANIZATION:

Linear structure containing adjustable arrays.

E. KEY IDENTIFIER (CONTROL FIELD):

FLDTAB is the major structure name. It is the name of an external variable containing the data.

F. RECORD LENGTH:

918 bytes(396 hexadecimal) minimum plus 10 (A hex) bytes per field name.

The minimum length includes the necessary PL/I dope vector (188 bytes = BC hexadecimal) and space for the RECLEN and keyname field names.

G. BLOCKING FACTOR:

Not Applicable

H. PURPOSE:

The field name table, FLDTAB, contains a list of the names of the fields in the current data base organized particularly for use by the DISPLAY and PRINT commands and index indicators, particularly for the EXPAND and SELECT commands, of field names for which there are inverted indexes. The sizes, base addresses and names of sequential format definition tables are tabulated. The base addresses and names of columnar format definition tables are tabulated.

```
**********************************************************************************************************
**********************************************************************************************************
                                                DATAPLEX = XXXXXX
**********************************************************************************************************
**********************************************************************************************************
**********************************************************************************************************
*F       A S I R S V B N V U I G       R       V       F   F   E   E    NAMEFLD  S       V                 *
* L      S U N E U A I U A N N E       E       A       L   L   L   L            E       A                 *
*  D     S B V A B R T M R I D  N      F       L       D   D   T   T   C  F     C       L                 *
*   N    O F F D C F F A E Q E   E     O       I       P   L   L   L   O  I     U       I                 *
*    A   C I I O N L L L L U X    R      R       D       O   E   E   I  D  E     R       D                 *
*     M  F L L N T D D I T E E     C      M       R       S   N   N   M  E  L     I       A                 *
*      E I E E L R    G   L X      R      A       T       I                D     T       R                 *
*        L    Y L    N   T T      T      T       N       T                      Y       G                 *
**********************************************************************************************************
 XXXXXXXX X X X X X X X X X X X XXXXXXXX XXXXXXXX XXXXXXXX XXXX XXXX XXX XXXX X.XXXXXXXX XXXXXXXX XXXXXXXXXXXXXXXXXX
                                                                  X.XXXXXXXX XXXXXXXX XXXXXXXXXXXXXXXXXX
                                                                  X.XXXXXXXX XXXXXXXX XXXXXXXXXXXXXXXXXX

*****************************************
*S F D    B    D S D M M L R     RSECTYCD  *
*U I E    S    E P A N N O E              *
*F L S     E   S A T T T A  C     C      M *
*I E C     L   C N A N N D  S      O     A *
*X T  R     N  O N   A I A   E      D     S*
*  Y  C     G  K E   B N B   C       E    K*
*  P   T     T   D   L G L    F           *
*  E         H       E   E    P           *
*****************************************
 X X XXXX XXXX X X X X X X XXXX XXXXXXXX:XX
                             XXXXXXXX:XX
```

FIGURE 1. SAMPLE LISTING FORMAT

1472

I. SCHEMATIC DIAGRAM:

See Figure 1

J. PL/I DECLARATION:

```
/*   FLDTAB:    NASIS   SYSTEM   FIELD    NAME   TABLE   FOR
     DATABASE-2.

     THIS  TABLE IS ALLOCATED (OR FREED AND REALLOCATED) AND
     INITIALIZED  BY A CALL  TO THE  ENTRY  POINT DBPFLDT OF
     MODULE FDBFAC.  EACH CALL  TO  THIS  ENTRY POINT CAUSES
     THE ENTIRE TABLE TO BE RE-INITIALIZED TO THE VALUES FOR
     THE CURRENTLY SPECIFIED DATABASE FILE.  THE VALUES WILL
     BE ADJUSTED TO REFLECT THE DATA AVAILABILITY BASED UPON
     THE SECURITY CODE ENTERED BY THE USER.                 */

 DECLARE
1 FLDTAB EXT CONTROLLED,          /*NASIS FIELD NAME TABLE */

  3 DATAPLEX CHARACTER(8),        /*THE DATABASE FILE NAME */

    3 FIELD,                      /*THE DATABASE FLD NAMES */
      5 # FIXED BINARY,           /*THE NUMBER OF FIELD    */
                                  /*NAMES IN THE TABLE     */
      5 RECLEN CHARACTER(8),      /*THE RECORD LENGTH FIELD*/
                                  /*ENTRY (FIELD.NAME(0))  */
      5 KEYNAME CHARACTER(8),     /*THE KEY FIELD ENTRY    */
                                  /*(FIELD.NAME(1))        */
      5 NAME(*) CHARACTER(8),     /*THE REMAINING FIELD    */
                                  /*NAMES FOR THIS FILE    */
                                  /*(FIELD.NAME(2:FIELD.#))*/
      5 INDEXID(*) CHARACTER(1),  /*THE INDEX FILE SUFFIX  */
                                  /*FOR EACH INDEXED FIELD */
                                  /*(FIELD.INDEX(0:FIELD.#))*/
      5 SUBFILE(*) CHARACTER(1),  /*THE SUBFILE SUFFIX FOR */
                                  /*EACH CONTROL FIELD OR  */
                                  /*SUBFILE FIELD          */
                                  /*(FLD.SUBFILE(0:FLD.#)) */

    3 SEQ_FORMAT(25),             /*SEQUENTIAL FORMAT INDEX*/
      5 # FIXED BINARY,           /*THE NUMBER OF FIELD    */
                                  /*NAMES IN THE FORMAT    */
      5 BASE POINTER,             /*THE FORMAT DESCRIPTION */
                                  /*TABLE ADDRESS          */
      5 NAME CHARACTER(8),        /*THE NAME ASSIGNED TO   */
                                  /*THIS FORMAT (OPTIONAL) */

    3 COL_FORMAT(25),             /*COLUMNAR FORMAT INDEX  */
      5 BASE POINTER,             /*THE FORMAT DESCRIPTION */
                                  /*TABLE ADDRESS          */
      5 NAME CHARACTER(8);        /*THE NAME ASSIGNED TO   */
                                  /*THIS FORMAT (OPTIONAL) */
```

K. FIELD DETAILS:

DATA BASE - has the name of the current dataplex.

FIELD. - the subscript in FIELD.NAME of the last field name. Thus it is the number of field names excepting RECLEN.

FIELD.RECLEN - the name of the anchor record length field. This may be referenced as FIELD.NAME(0) when convenient.

FIELD.KEYNAME - the name of the current data base's anchor key field. This may be referenced as FIELD.NAME(1) when convenient.

FIELD.NAME - an array containing the names of the fields in the current data base arranged as shown in Paragraph I.

FIELD.INDEXID - An array parallel with FIELD.NAME. A non-blank indicates an indexed field.

FIELD.SUBFILE - An array parallel with FIELD.NAME. In the anchor and associated portion of the array (subscripts 0 through FLDTAB.SEQ.FORMAT (3)) a non-blank indicates a subfile control field. In the subfile portion of the array(subscripts above FLDTAB.SEQ-FORMAT(3)) the character indicates which subfile a field is on.

SEQ_FORMAT - an array serving as a directory of the sequential format definition tables. The first four entries are posted by RDBPAC to overlay FIELD.NAME beginning with FIELD.KEYNAME as shown in Paragraph I. The remaining entries are posted by RIBFORM to refer to dynamically allocated sequential format definition tables.

SEQ_FORMAT. - the number of field names in a sequential format definition table.

SEQ_FORMAT.BASE - the address of a sequential format definition table or a NULL pointer value if it is undefined.

SEQ_FORMAT.NAME - the name assigned to a sequential format or blanks.

COL_FORMAT - an array serving as a directory of the columnar format definition tables. The entries are posted by RDBFCRM to refer to dynamically

allocated columnar format definition tables.

COL_FORMAT.BASE - the address of a columnar format definition table or a NULL pointer value if it is undefined.

COL_FORMAT.NAME - the name assigned to a columnar format or blanks.

	SEQ-FORMAT .	.BASE
1	1	
Predefined 2	5	
3	8	
4	16	
5		
User defined		
25		

	FIELD NAME	.INDEX	.SUBFILE		
0	RECLEN				
1	EMPNO				KEY FIELD
2	AREACODE				
3	EMPAGE	A			ANCHOR FIELDS
4	KIDCTL		Z	*	
5	PHONENUM				
6	ADDRESS				
7	CARCTL		Y	*	ASSOCIATED
8	HIREDATE	B			FIELDS
9	KIDID		Z	**	
10	CNAME	C	Z		
11	KIDPK		Z		
12	PETS	D	Z		
13	CARID		Y	**	SUBFILE FIELDS
14	CARPK		Y		
15	MAKE	E	Y		
16	WEIGHT	F	Y		

Alphabetic Sequences

* Subfile Control Fields

** Subfile id Key Fields

Figure 1. Schematic Diagram of FLDTAB

TOPIC F.11 - DATA RETRIEVAL

A. DATA SET NAME:

FORMATS Display Format

B. CREATED BY:

Formats - RDBSTRT

C. TYPE OF FILE:

Terminal Display (Pageable)

D. ORGANIZATION:

Not Applicable

E. KEY IDENTIFIER (CONTROL FIELD):

Not Applicable

F. RECORD LENGTH:

Not Applicable

G. BLOCKING FACTOR:

Not Applicable

H. PURPOSE:

This terminal display is created to display the names of the formats currently available to him. A title, identifying the display, is generated, followed by the format names. The eight character names are sorted into alphabetic sequence, tagged with an asterisk if the format is in core, separated by a blank and grouped into a SCRNWTH size line before they are written to the display.

TOPIC F.12 - DATA RETRIEVAL

A. DATA SET NAME:

SETAB Sets Table

B. CREATED BY:

Not Applicable

C. TYPE OF FILE:

Table

D. ORGANIZATION:

PL/I Data Structure

E. KEY IDENTIFIER (CONTROL FIELD):

Not Applicable

F. RECORD LENGTH:

Not Applicable

G. BLOCKING FACTOR:

Not Applicable

H. PURPOSE:

1. SETAB EXTERNAL CONTROLLED.
This structure contains the sets, i.e., current strategy, that the user is creating and associated information.

2. CURRENT_# BINARY FIXED (15,0).
This is the value of the last set number that was created.

2. SET (0:99).
There is one set created for each select, search and LIMIT COMMAND.

3. POINTER (97) POINTER.
There is a pointer to a list of keys for every set that is created. POINTER (J) points to the list for SRT (J).

3. SIZE BINARY FIXED (31,0).
This is the number of keys associated with

the corresponding set number.

3. TYPE CHARACTER (1).
This is the SUBFILE SUFFIX that describes the origin of the keys in the set.

I. PL/I DECLARATION:

```
/*                  NASIS SYSTEM SET TABLE                        */

DCL 1 SETAB EXTRNL CONTROLLED,/*DEFINE THE SET TABLE      */
 2 CURRENT_# BIN FIXED(15,0), /*LAST ASSIGNED SET NUMBER */
 2 SET(0:99),                 /*DEFINE THE SET ENTRIES    */
  3 POINTER PTR,              /*THE SET LIST POINTER      */
  3 SIZE BIN FIXED(15,0),     /*THE SET SIZE (# OF KEYS)  */
  3 TYPE CHAR(1);             /*THE SET TYPE (SUBFILE ID) */
```

TOPIC F.13 - DATA RETRIEVAL

A. DATA SET NAME:

USERTAB User Data Table

B. CREATED BY:

RDBMTT

C. TYPE OF FILE:

Table

D. ORGANIZATION:

PL/I Data Structure

E. KEY IDENTIFIER (CONTROL FIELD):

Not Applicable

F. RECORD LENGTH:

Not Applicable

G. BLOCKING FACTOR:

Not Applicable

H. PURPOSE:

1. USERTAB EXTERNAL CONTROLLED.
This structure contains user oriented and status information useful to all NASIS sub-systems.

2. NASIS_ID CHARACTER(8) VARYING.
This field contains the id specified by the user when initiating his NASIS session.

2. SECURITY CHARACTER (8) VARYING.
This field contains the user's most recently specified security code, i.e. his PASSWORD at logon or in respond to a SECURE Command.

2. OWNER_ID CHARACTER (8) VARYING.
This field contains the TSS userid of the owner of the file specified by the user.

2. STRATEGY CHARACTER (16) VARYING.
This field contains the name of the strategy in the event of a RERUN.

2. TASK_ID BINARY FIXED (31,0).
This field contains the task identification number assigned to the user at logon time.

2. SEQUENCE BINARY FIXED (15,)).
This field contains a sequence number used by the system in defining unique ddnames to dynamically specified files.

2. BITS.
The status of the user's session is reflected by the settings of the following bit switches.

3. MTTFLAG BIT(1).
Describes whether the task is running under MTT or not.

3. DISABLED BIT(1).
Defines the status of attention interupts.

3. RETRIEVR BIT(1).
Describes whether the task is running under the retrieval system or not.

3. RESTART BIT(1).
Describes whether the session is a restart.

3. RERUN BIT(1).
Describes whether the session is a rerun.

3. TESTMODE BIT(1).
Describes whether the session is productive or a debugging run.

3. CONVFLAG BIT(1).
Describes whether the task is conversational or not.

I. PL/I DECLARATION:

```
/*                NASIS SYSTEM USER DATA TABLE                */

DCL
1 USERTAB EXT CONTROLLED,       /*NASIS USER DATA TABLE       */
  2 NASIS_ID CHAR(8) VAR,       /*USER'S IDENTIFICATION       */
  2 SECURITY CHAR(8) VAR,       /*USER'S SECURITY CODE        */
  2 OWNER_ID CHAR(8) VAR,       /*FILE OWNER'S IDENTIFIER     */
  2 STRATEGY CHAR(16) VAR,      /*STRATEGY NAME FOR RERUN     */
  2 TASK_ID BIN FIXED(31,0),    /*TASK IDENTIFICATION #       */
```

```
2 SEQUENCE BIN FIXED(15,0), /*DDNAME SEQUENCE NUMBER     */
2 BITS,                     /*SYSTEM STATUS FLAGS        */
  3 MTTFLAG  BIT(1),        /*'1'=IN MTT MODE            */
  3 DISABLED BIT(1),        /*'1'=ATTN'S DISABLED        */
  3 RETRIEVE BIT(1),        /*'1'=RUNNING RETRIEVAL      */
  3 RESTART  BIT(1),        /*'1'=IN RESTART MODE        */
  3 RERUN    BIT(1),        /*'1'=IN RERUN MODE          */
  3 TESTMODE BIT(1),        /*'1'=NO STRATEGY SAVING     */
  3 CONVFLAG BIT=1),        /*'1'=CONVERSATIONAL         */
  3 EXTRABIT BIT(1);        /*'1'=(NO ASSIGNED VALUE)    */
```

3. X is actual value.

TOPIC F.14 - DATA RETRIEVAL

A. DATA SET NAME:

EXPLAIN Display Format

B. CREATED BY:

EXPLAIN (message, RESPONSE and term options) - RDBEXPL

C. TYPE OF FILE:

Terminal Display (Pageable)

D. ORGANIZATION:

Not Applicable

E. KEY IDENTIFIER (CONTROL FIELD):

Not Applicable

F. RECORD LENGTH:

Not Applicable

G. BLOCKING FACTOR:

Not Applicable

H. PURPOSE:

This terminal display is created to display to the user the results of his message response or term explanation. The date will be written to the screen as read from the message file with no indentation or data tagging, but with word-break specified.

TOPIC F.15 - DATA RETRIEVAL

A. DATA SET NAME:

GFIELDS Display Format

B. CREATED BY:

GFIELDS - RDBGFLDS

C. TYPE OF FILE:

Terminal Display

D. ORGANIZATION:

Not Applicable

E. KEY IDENTIFIER (CONTROL FIELD):

Not Applicable

F. RECORD LENGTH:

Not Applicable

G. BLOCKING FACTOR:

Not Applicable

H. PURPOSE:

This terminal display is created to display the names assigned to the various generic levels of the key of the generic file. A title, identifying the display, is generated, followed by the names. The eight character names,separated by two blanks, are grouped into SCRNWTH size lines before they are written to the display.

TOPIC F.16 - DATA RETRIEVAL

A. DATA SET NAME:

SEQ_FORM - Sequential Format Definition Table

B. CREATED BY:

DBPFLDT entry of RDBPAC (formats 1-4) overlay FLDTAB.FIELD.NAME

RDBFORM (formats 5-25) - by the FORMAT command.

C. TYPE OF FILE:

(4) Table

D. ORGANIZATION:

Adjustable linear array of 8-character field names.

E. KEY IDENTIFIER (CONTROL FIELD):

SEQ_FORM is the major structure name.

F. RECORD LENGTH:

8 bytes times the number of field names in FLDTAB.

G. BLOCKING FACTOR:

Not Applicable

H. PURPOSE:

A sequential format definition table, SEQ_FORM, contains a list of the names of the fields in a sequential format for use by the REVIEW and RECORD commands. The key field name is always the first in the list. The number of names in the list and the base address of the list is posted in FLDTAB.SEQ_FORMAT.

I. PL/I DECLARATION:

```
DECLARE
  1 SEQ_FORM BASED(SEQ_BASE),/* SEQUENTIAL FORMAT SPECS   */
    3 FIELD_1_,              /* OVERLAID BY FIELD(1)      */
      5 #FIELDS FIXED BIN,   /* NOT USED                  */
      5 PAD CHAR(6),         /* FILLER TO CHAR(8)         */
    3 FIELD(2:I
      REFER(SEQ_FORM.#FIELDS)),/* NOT USED                */
      5 NAME CHAR(8);        /* NAME LIST                 */
```

TOPIC F.17 - DATA RETRIEVAL

A. DATA SET NAME:

NASISID.STRATEGY.DATASET

B. CREATED BY:

RTSSTRT

C. TYPE OF FILE:

(5) Non-Data Base

D. ORGANIZATION:

VISAM

E. KEY IDENTIFIER (CONTROL FIELD):

Strategy name supplied by the user (region name) plus a seven digit integer generated key.

F. RECORD LENGTH:

256 Bytes

1. Key length - 23
   a. Region name - 16
   b. Integer key - 7
2. Data length - 231
3. Record length field - 4
4. Keyboard/Cardboard Indicator - 1

G. BLOCKING FACTOR:

Not Applicable

H. PURPOSE:

This data set is used to contain the command strings that the user has entered through the NASIS system. The current strategy is kept in a CURRENT_STRATEGY as each command is entered. After a strategy session, the user may save the current strategy under his specified name; otherwise, it will be deleted. Tese saved strategies can then, at some later time, be rerun by

use of the RERUN command. The saving of the current strategy command strings also provides for a restart capability if TSS crashes while the user is running the NASIS system.

TOPIC F.18 - DATA RETRIEVAL

A. DATA SET NAME:

SRCHTAB - Linear Search Table of Pseudo-sets

B. CREATED BY:

SELECT (search option) - RDBSLCT

C. TYPE OF FILE:

Table

D. ORGANIZATION:

Linear structure containing arrays.

E. KEY IDENTIFIER (CONTROL FIELD):

SRCHTAB is the major structure name; it is the name of the control section containing the data.

F. RECORD LENGTH:

(hexadecimal) data bytes.

G. BLOCKING FACTOR:

Not Applicable

H. PURPOSE:

The linear search table of pseudo-sets, SRCHTAB, contains the displayable information for every pseudo-set, whether a PRINT is to be performed on each, the set upon which the pseudo-set is based (as a result of the search), pointers to search lists and parsing lists (for those pseudo-sets defined as boolean combinations of other pseudo-sets and/or sets), and the testing criteria for each.

I. SCHEMATIC DIAGRAM:

See Figure 1

J. VARIABLE DETAILS:

3 CURRENT_S# is the last pseudo-set number assigned.

3 MAX_S is the maximum allowable pseudo-set number.

3 ENTRY is an array of displayable pseudo-set information containing the S# and its related EXPRESSION.

3 ENTRYDEF is an array with detailed pseudo-set information:

5 RECORD is a pointer used for the parameters of those pseudo-sets to be RECORDed after a search execution. This points to SPRNTAB structure.

5 CREATED_BY is two bits identifying the type of SELECT command used to create this pseudo-set where,

7 SELECT_IF bit is on if the search option was used, or

7 SELECT_BOOL bit is on if the boolean option was used.

5 REF_SET is a structure used for identifying the searching universe (or set) wherein,

7 PTR is pointer to set to search within

7 S# is a bit on if the set to be searched is a pseudo-set.

5 CORRES_SET# is the value of the set resulting from this pseudo-set.

5 LIST_PNTR is a pointer to the search list structure for this pseudo-set.

5 PARSED is a pointer to a parsing structure for boolean-created pseudo-sets.

5 DELETED is a bit on if this pseudo-set has been deleted.

5 FIELDNAM is the field name to be tested.

5 OP_CODE is a value of the operator to be used for the test, as follows:

1. greater than
2. less than
3. equal
4. greater than or equal

```
                5.  less than or equal
                6.  not equal
                7.  between
                8.  containing

            5   VALUES is a pointer to test values; this
                points to VALUTAB.

        3   SRCH_IN_PROGRESS is a bit on if a search is being
            executed.

        3   IFSO is set on if any pseudo-set is to be
            printed.

        3.  SLCT_ERROR is a bit on if error occurs in SELECT
            during execution of search.

        3.  SLCT_FINISH is a bit on if all SELECT functions
            are complete during execution of search.

        1.  S#_XREFS is a bit array used to record the S#'
            which reference the S# in question.

        1.  SPRNTAB is a table of record parameters of an
            S#.

            3.  FORMAT is a table of record format
                parameters.

                5.  TYPE


                5.  FIRST 5.  LAST

            3.  NEXT_SPRNTAB is a pointer to next SPRNTAB
                structure.

        1.  VALUE_# is set for adjustment of valutab at
            allocation.

        1.  VALUTAB is a table of pointers to values.

            3.  #OF is number of pointers in this table.

            3.  VALUPTR is an array of pointers to values.

        1.  VALUE_SI E is set for size of value at
            allocation.

        1.  VALUE is a table containing a value to be used
            during search.  Pointer is in VALUTAB.

            3.  SI E is length of value.
```

Page intentionally left blank

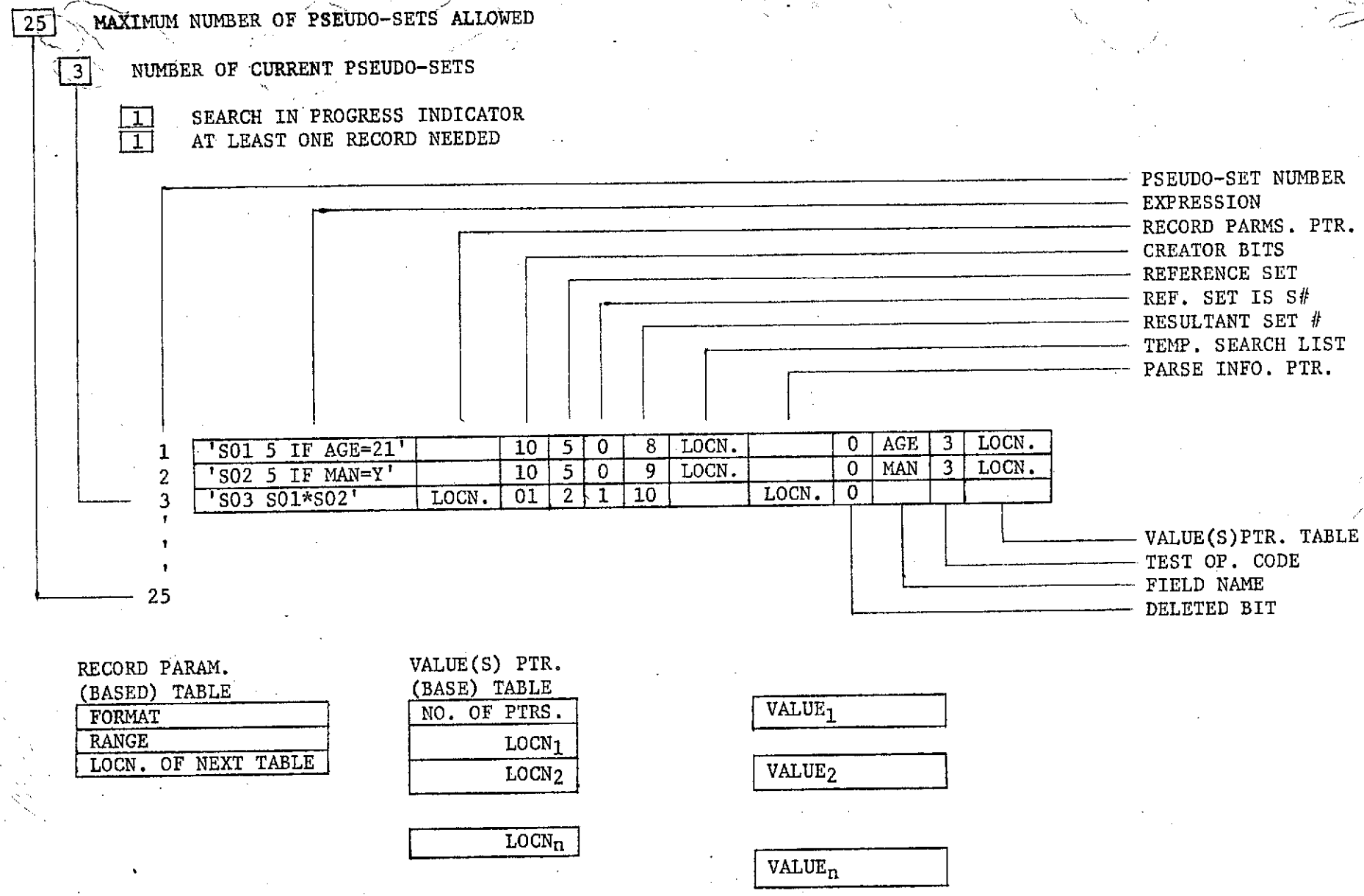


Figure 1. SCHEMATIC DIAGRAM

TOPIC F.19 - DATA RETRIEVAL

A. DATA SET NAME:

COL_FORM - Columnar Format Definition Table

B. CREATED BY:

RDBFORM - the FORMAT command

C. TYPE OF FILE:

(4) Table

D. ORGANIZATION:

Structure containing miscellaneous items, a linear array, and an adjustable array of structures.

E. KEY IDENTIFIER (CONTROL FIELD):

COL_FORM is the major structure name.

F. RECORD LENGTH:

1340 bytes plus 40 bytes per field specification, i.e., 1540 bytes minimum (5 field specifications) to 3980 bytes maximum (66 field specifications).

G. BLOCKING FACTOR:

Not Applicable

H. PURPOSE:

A columnar format definition table, COL_FORM, contains coded specifications for a columnar display. It is used by the DISPLAY and PRINT commands and may be revised by the FORMAT command. The optional line for the page number, lines for titles, and lines for headers hold literal text for output. For each field specified, the field name, column, width, summary requirements, tally, and summation values are carried. (Average is not carried in COL_FORM; it is computed in DISPLAY or PRINT.)

I. PL/I DECLARATION:

```
DECLARE
 1 COL_FORM BASED(COL_BASE),   /*COLUMNAR FORMAT SPECS   */
   3 LINESIZE FIXED BIN(31),   /*SCRNCOL OR 132         */
   3 RECORD_COUNT FIXED BIN(31),/*INIT(0)               */
```

```
3 TOP,
  5 (PAGE#,                    /*1 OR 0 LINES                 */
     #TITLES,                  /*0 OR MORE LINES              */
     #HEADERS,                 /*0 OR MORE LINES              */
     DEFAULT_HDR)FIXED BIN,/*0 OR RELATIVE HEADER LINE*/
  5 LINE(10) CHAR(132),
3 COL_GIVEN BIT(1),            /*1: FIELD COLUMNS GIVEN  */
                               /*0: FIELD COLUMNS DEFALTD*/
3 #FIELDS FIXED BIN,
3 FIELD( I REFER(CCI_FORM.#FIELDS)),
  5 NAME CHAR(8),
  5 COLUMN FIXED BIN,          /*FOR TRUNCATION INDICATOR*/
                               /*USE COLUMN+1...FOR VALUE*/
  5 WIDTH FIXED BIN,           /*WITHOUT                      */
                               /*TRUNCATION INDICATOR         */
  5 ELEMENT_LIMIT FIXED BIN,/*FOR RETRIEVAL                 */
  5 ELEMENT_TALLY,
    7 REQUIRED BIT(1),         /*INIT('0'B)                   */
    7 # FIXED BIN(31),         /*INIT(0)                      */
  5 ELEMENT_SUM,
    7 REQUIRED BIT(1),         /*INIT('0'B)                   */
    7 ZONED BIT(1),            /*1: ZONED VALUE (INIT)        */
                               /*0: BINARY VALUE              */
    7 VALUE FLOAT BIN(53),     /*INIT(0)                      */
  5 ELEMENT_AVERAGE_REQUIRED
    BIT(1);                    /*INIT('0'B)                   */
```

TOPIC F.20 - DATA RETRIEVAL

A. DATA SET NAME:

FIELDS Display Format

B. CREATED BY:

FIELDS - RDBFLDS

C. TYPE OF FILE:

Terminal Display (Pageable)

D. ORGANIZATION:

Not Applicable

E. KEY IDENTIFIER (CONTROL FIELD):

Not Applicable

F. RECORD LENGTH:

Not Applicable

G. BLOCKING FACTOR:

Not Applicable

H. PURPOSE:

This terminal display is created to display the names of the fields available to him from the current file. A title, identifying the display, is generated, followed by the field names. The eight character names, flagged by an asterisk, if indexed, and separated by a blank, are grouped into SCRNWTH size lines before they are written to the display. As each subfile is encountered, a heading, identifying it and its control field, is generated.

TOPIC F.21 - RETRIEVAL

A. DATA SET NAME:

LIMIT

B. CREATED BY:

The module whose name is formed by concatenating "RL" to the data base name i.e. "RLERTS". The writeup on creating this module is in the DBA User's Guide.

C. TYPE OF FILE:

table.

D. ORGANIZATION:

PL/I Data Structure

E. KEY IDENTIFIER (CONTROL FIELD):

Not Applicable

F. RECORD LENGTH:

Not Applicable

G. BLOCKING FACTOR:

Not Applicable

H. PURPOSE:

The LIMIT structure contains the anchor key subfield names, their starting position within the key, and their length. This table is used by the LIMIT command when limiting a set.

I. PL/I DECLARATION:

```
DECLARE
1 LIMIT              EXTERNAL CONTROLLED, /* DEFINE THE TAB*/
    3 KEY SIZE       BIN FIXED, /* LENGTH OF FORMATTED KEY.*/
    3 #_ENTRIES      BIN FIXED, /* NUMBER OF SUBFIELDS     */
                                /* DEFINED ON THE ANCHOR KEY. */
    3 FIELD (16),               /* ONE ENTRY FOR EACH SUBFIELD*/
      5 NAME         CHAR (8) VAR, /* NAME OF SUBFIELD.    */
      5 START        BIN FIXED, /* START OF SUBFIELD,      */
                                /* WHERE 1 = FIRST CHARACTER  */
                                /* POSITION OF THE KEY.       */
      5 LNGTH        BIN FIXED, /* LENGTH OF SUBFIELD.     */
      5 TEST         BIT (1), /* WHETHER OR NOT TO APPLY   */
```

```
                        /* THIS TEST WHEN LIMIT IS     */
                        /*    CALLED.                  */
5 VALUE (2) CHAR (50) VAR;
                        /* THE VALUES TO COMPARE A     */
                        /* KEY SUBFIELD AGAINST.       */
```

TOPIC F.22 - RETRIEVAL

A. DATA SET NAME:

LIMIT Command Display Format

B. CREATED BY:

RDBLMT

C. TYPE OF FILE:

Terminal Display

D. ORGANIZATION:

Not Applicable

E. KEY IDENTIFIER (CONTROL FIELD):

Not Applicable

F. RECORD LENGTH:

Not Applicable

G. BLOCKING FACTOR:

Not Applicable

H. PURPOSE:

This display allows the user to view the new set created by LIMIT.

I. SAMPLE DISPLAY:

```
SET#              XRETS          EXPRESSION          PAGE 1
XX                XXXX           XXXXXXXXXXXXXXXXXXXXXX
```

TOPIC G.1 - USAGE STATISTICS

A. DATA SET NAME:

STATIC Data Set Descriptors

B. CREATED BY:

Command System and Maintenance System

C. TYPE OF FILE:

Dataplex

D. ORGANIZATION:

VISAM

E. KEY IDENTIFIER (CONTROL FIELD):

Not Applicable

F. RECORD LENGTH:

4000/V

G. BLOCKING FACTOR:

Not Applicable

H. PURPOSE:

Maintain system statistics:

1. Retention of Statistics

In order to maintain the usage statistics, a file is required; and, in order to enhance the usage of these statistics (to interface smoothly with the NASIS system), a data base is required. With the statistics on a data base, the full power of the NASIS system can be used to manipulate them. If any special retrieval or report programs are required, then, currently, DBPL/I and TSPL/I are available; and the Report Generator and Linear Search features are also available.

Each TSS-ID joined to the NASIS system will have its own statistics data base, and it can be shared with other TSS-ID's just as any other data base.

The name of the statistic data base will follow a uniform format for each TSS-ID joined to NASIS. That is, the name of the statistic data base will be STATIC and the fully qualified data base name will follow the standard naming conventions, eg.

tss-id.STATIC.STATIC#
tss-id.STATIC.STATIC

for the descriptors and anchor file, respectively.

This design also facilitates the integrity and security of the statistic file in that only the owner or those permitted by the owner can access the file.

2. Accumulation of Statistics

The STATIC data base will be composed of two different record types. The data required, and how it will be kept, is as follows:

a. KEY

SEPARATE - A single character which will distinguish the record type. A value of zero will indicate a maintenance record. A value of one will indicate a retrieval record.

IDENTIFIER - For maintenance records, it will be the data base name padded with dollar signs. For retrieval records, it will be the NASIS-ID padded with asterisks (to eight characters). Appended to the NASIS-ID will be the data base owner's TSS-ID and the name of the inverted index file.

b. The maintenance fields are as follows:

TOTALTRN - the number of transactions processed.

ANCOUNT - the number of records on the anchor file.

TOTALRUN - the number of maintenance runs.

MAINDATE - the date of each maintenance run:

element 1 will be the dates of the data base creation, elements 2 through 13 will be the dates of individual maintenance runs. After the dates have been filled, the second one will be dropped and the newest date added on the end. This field is variable-length with fixed-length elements. There is a maximum of thirteen elements.

TRANCNEW

TRANCDEL

TRANCUPD

TRSUBNEW

TRSUBDEL

TRSUBUPD

TRINVNEW

TRINVDEL

TRINVUPD

- where TR indicates transaction; ANC, the anchor file; SUB, the subrecord files; INV, inverted files; NEW, new records; DEL, deletions; and UPD, updates.

These are the transaction count fields required for maintenance statistics. These fields will be used in conjunction with the data field. They will also have a maximum of thirteen elements. The elements will correspond directly with the date field and will represent the number of that given type of transaction encountered during the maintenance run. When all of the elements are present, the next count inserted will cause the second count to be added to the first element and the second element dropped. The newest element will go on the end.

c. The retrieval fields are:

CONNTIME - the connect time

CPUTIME - the CPU time

TOTALES - number of sessions

STRATLEN - the strategy length

STRATSTR - the number of strategies stored

STARDTE - the date of the first terminal session

LASTDATE - the date of the last terminal session

STRATNME - the names of the stored strategies - maximum of four.

NOTE: The eight fields above are to be accumulated for each NASIS-ID. There may be many records for each NASIS-ID; therefore, these statistics will be kept in a special record. The OWNER-ID and the inverted file name in this special record will be equal to blanks.

#EXPANDS - number of EXPANDS per session

#SELECTS - number of SELECTs per session

#SEARCHS - number of SEARCHes per session

#CORECTS - number of CORRECTs per session

SESSDATE - the date of each session

These field all contain a maximum of thirteen elements. The first element represents an accumulator and contains the total for all occurrences up to the SESSDATE, which is the date of the last ejected session of the list (the earliest session). Regardless of the actual number of sessions within one calendar day, the statistics will be accumulated as if there were only one session.

All of the maintenance statistics will be automatically updated with the Load/Create program and the Maintenance Mainline program. If the data base owner wishes to modify certain data pertaining to the maintenance statistics, he has the ability to use the CORRECT command to update the STATIC data base interactively.

All cf the retrieval statistics will be automatically updated with the FINISH module of the command system. If required, at maintenance time, a 'snapshot' of the statistics will be printed. If the data base owner (system manager) wishes to modify certain data pertaining to the retrieval statistics, he has the ability to use the CORRECT command to interactively update the STATIC data base.

APPENDIX A.

The STATIC data base is composed of the following fields:

A. KEY

1. Alphanumeric.
2. Fixed field.
3. Length of 24 bytes.
   a. First byte is maintenance or retrieval record indicator.
      1. 0 = maintenance record.
         a. data base name left justified.
         b. remainder padded with '$'s.
      2. 1 = retrieval record.
         a. NASIS-ID//OWNER-ID//DATA BASE file-name.
            1. The NASIS-ID is eight characters long and padded with '*'s.
            2. The OWNER-ID is really a TSS-ID, eight characters long and padded with '*'s.
            3. The data base file-name is seven characters long and conforms to the data base - dataset naming conventions.

B. TOTALTRN (Maintenance)
1. Alphanumeric
2. Fixed field
3. Length of 6 bytes
4. Contains the total number of transactions.

C. ANCOUNT (Maintenance)
1. Alphanumeric
2. Fixed field.
3. Length of 6 bytes.
4. Contains number of records on the anchor file.

D. TOTALRUN (Maintenance)
1. Alphanumeric.
2. Fixed field.
3. Length of 3 bytes.
4. Contains the number of maintenance runs.

E. MAINDATE (Maintenance)
1. Alphanumeric.
2. Fixed element.
   a. Total of 13 elements, each 6 bytes long.
   b. In the form MM/DD/YY to indicate the month, day, and year of each maintenance run.

Element 1 will contain the data base creation date while elements 2-13 will be the dates of the individual maintenance runs. After the dates have been filled, the second one will be dropped and the newest date added to the end.

3. Total length of 78 bytes.

F. TRANCNEW (Maintenance)
   1. Alphanumeric.
   2. Fixed elements.
      a. Each 6 bytes long.
      b. Maximum of 13 elements.
   3. Total length of 78 bytes.

G. TRANCDEL (Maintenance)
   1. Alphanumeric.
   2. Fixed element.
      a. Each 6 bytes long.
      b. Maximum of 13 elements.
   3. Total length of 78 bytes.

H. TRANCUPD (Maintenance)
   1. Alphanumeric.
   2. Fixed element.
      a. Each 6 bytes long.
      b. Maximum of 13 elements.
   3. Total length of 78 bytes.

I. TRSUBNEW (Maintenance)
   1. Alphanumeric.
   2. Fixed element.
      a. Each 6 bytes long.
      b. Maximum of 13 elements.
   3. Total length of 78 bytes.

J. TRSUBDEL (Maintenance)
   1. Alphanumeric.
   2. Fixed element.
      a. Each 6 bytes long.
      b. Maximum of 13 elements.
   3. Total length of 78 bytes.

K. TRSUBUPD (Maintenance)
   1. Alphanumeric.
   2. Fixed element.
      a. Each 6 bytes long.
      b. Maximum of 13 elements.
   3. Total length of 78 bytes.

L. TRINVNEW (Maintenance)
   1. Alphanumeric.
   2. Fixed element.

a. Each 6 bytes long.
b. Maximum of 13 elements.
3. Total length of 78 bytes.

M. TRINVDEL (Maintenance)
1. Alphanumeric.
2. Fixed element.
a. Each 6 bytes long.
b. Maximum of 13 elements.
3. Total length of 78 bytes.

N. TRINVUPD (Maintenance)
1. Alphanumeric.
2. Fixed element.
a. Each 6 bytes long.
b. Maximum of 13 elements.
3. Total length of 78 bytes.

NOTE: Items F through N are transaction count fields for the maintenance statistics and correspond directly to MAINDATE.

TR indicates TRANSACTION
ANC indicates ANCHOR FILE
INV indicates INVERTED FILE
NEW indicates NEW RECORDS
DEL indicates DELETIONS
UPD indicates UPDATES

O. CONNTIME (Retrieval)
1. Alphanumeric.
2. Fixed field.
3. Length of 10 bytes.
4. Contains connect time.

P. CPUTIME (Retrieval)
1. Alphanumeric.
2. Fixed field.
3. Length of 10 bytes.
4. Contains total CPU time.

Q. TOTALSES (Retrieval)
1. Alphanumeric.
2. Fixed field.
3. Length of 10 bytes.
4. Contains total number of sessions.

R. STRATLEN (Retrieval)
1. Alphanumeric.
2. Fixed field.
3. Length of 6 bytes.
4. Contains strategy length.

S. STRATSTR (Retrieval)
   1. Alphanumeric.
   2. Fixed field.
   3. Length of 1 byte.
   4. Contains number of strategies stored.

T. STRATNME (Retrieval)
   1. Alphanumeric.
   2. Fixed element.
      a. Each 35 bytes long.
      b. Maximum of 4 elements.
   3. Total length of 140 bytes.
   4. Contains names of stored strategies.

U. STARTDTE (Retrieval)
   1. Alphanumeric.
   2. Fixed field.
   3. Length of 6 bytes.
   4. Contains date of first terminal session.

V. LASTDATE (Retrieval)
   1. Alphanumeric.
   2. Fixed field.
   3. Length of 6 bytes.

NOTE: The right fields above are accumulated for each NASIS-ID. The owner-ID and the file-name have no meaning.

Therefore, the KEY of the record where these statistics are meaningful will be composed of an owner-ID and a file-name which are blank.

W. #EXPANDS (Retrieval)
   1. Alphanumeric.
   2. Fixed element.
      a. Each 6 byters long.
      b. Maximum of 13 elements.
   3. Total length of 78 bytes.

X. #SELECTS (Retrieval)
   1. Alphanumeric.
   2. Fixed element.
      a. Each 6 bytes long.
      b. Maximum of 13 elements.
   3. Total length of 78 bytes.

Y. #SEARCHS (Retrieval)
   1. Alphanumerics.
   2. Fixed element.
      a. Each 6 bytes long.
      b. Maximum of 13 elements.
   3. Total length of 78 bytes.

Z. #CORECTS (Retrieval)
   1. Alphanumerics.
   2. Fixed element.
      a. Each 6 bytes long.
      b. Maximum of 13 elements.
   3. Total length of 78 SESSDATE (Retrieval)

AA. SESSDATE (Retrieval)
   1. Alphanumerics.
   2. Fixed element.
      a. Each 6 bytes long.
      b. Maximum of 13 elements.
   3. Total length of 78 bytes.

NOTE: In the last 5 fields there is a one for one correspondence in the elements.

first SESSDATE - the date of the newest session in the accumulated counts.

first (others) - the accumulated counts on all indicated.

Regardless of the actual number of sessions within one given calendar day, the statistics will be accumulated as if there were only one session.

When (during UPDATE) a record is encountered with the variable fields having all 13 elements filled, the 'snapshot' of the given record will be taken. The last 12 elements will then be cleared, by summing them and adding them to the first element, the first element SESSDATE will be made equal to the last element SESSDATE.

TOPIC G.2 - USAGE STATISTICS

A. DATA SET NAME:

Maintenance Statistics Report Format

B. CREATED BY:

STATIC Report (RDEPRNTM)

C. TYPE OF FILE:

VS (print)

D. ORGANIZATION:

Sequential

E. KEY IDENTIFIER (CONTROL FIELD):

Not Applicable

F. RECORD LENGTH:

133 Bytes

G. BLOCKING FACTOR:

Not Applicable

H. PURPOSE:

To display (via listing) the status of the maintenance statistics.

MAINTENANCE STATISTICS FOR SYSTEMS MANAGER ** 01/11/73 PAGE 1

DATAPLEX NAME	TOTAL TRNS	ANCHOR RECORDS	NUMBER RUNS	TRANS RUN	MAINTENANCE DATES	FILEPLEX ADDS	FILEPLEX DELETES	FILEPLEX UPDATES	SUBPLEX ADDS	SUBPLEX DELETES	SUBPLEX UPDATES	XPLEX ADDS	XPLEX DELETES	XPLEX UPDATES
ASRD1$		3,132	1		12/19/72	3,132								

FILEPLEX	ADDS	DELETES	UPDATES	
TOTAL	3,132			FOR ALL RUNS
AVERAGE	3,132			PER RUN

781

TOPIC G.3 - USAGE STATISTICS

A. DATA SET NAME:

Retrieval Statistics Report Format

B. CREATED BY:

Report Print (RDBPRNTR)

C. TYPE OF FILE:

VS (print)

D. ORGANIZATION:

Sequential

E. KEY IDENTIFIER (CONTROL FIELD):

Not Applicable

F. RECORD LENGTH:

133 Bytes

G. BLOCKING FACTOR:

Not Applicable

H. PURPOSE:

To display (via listing) the status of the retrieval statistics.

RETRIEVAL STATISTICS 01/03/73

NASISID	CONN-TIME HR:MM:SC	CPU-TIME HR:MM:SC:MS	# SES	STRAT LENGTH	STORED #	OWNER ID	FILE NAME	FIELD NAME	ACTUAL EXP	TOTAL SEL	NUMBER OF SRCH	NUMBER OF CORR
NE01	0:53:30	0:00:48:790	5	0	0							
						SAOWNER	ASRD1$A	AUTHOR	3	0	0	0
						SAOWNER	ASRD1$B	KEYWORDS	13	0	0	0
						SAOWNER	DB2TDBA	EMPAGE	1	0	0	0
						SAOWNER	DB2TDBB	TOTALCAR	1	0	0	0
						SAOWNER	DB2TDBC	KIDAGE	1	0	0	0
						SAOWNER	DB2TDBD	PET	1	0	0	0
						SAOWNER	DB2TDBE	SVCDATE	1	0	0	0

TOPIC G.4 - USAGE STATISTICS

A. DATA SET NAME:

Snapshot Statistics Report Format

B. CREATED BY:

Snapshot Print (BDECHKPT)

C. TYPE OF FILE:

VS (print)

D. ORGANIZATION:

Sequential

E. KEY IDENTIFIER (CONTROL FIELD):

Not Applicable

F. RECORD LENGTH:

133 Bytes

G. BLOCKING FACTOR:

Not Applicable

H. PURPOSE:

To display (via listing) those records which have undergone the reinitialization process.

SNAPSHOT (CHECKPOINT)OF RETRIEVAL STATISTICS RECORDS BEFORE REINITIALIZATION

12/18/72

LISR ID	CONN-TIME HR:MIN:SC	CPU-TIME HR:MN:SC:MS	# SES	STRAT LENGTH	OWNER-ID	FIELD NAME	FILE NAME	SESSION DATE	# EXPANDS	# SELECTS	# SEARCHS	# CORRECTS
NEO1	:19:40	0:00:12:399	2		SAOWNER	KEYWORDS	ASRDI$B	721215				
								721215		1		
								721215				
								721215				
								721215				
								721215				
								721215				
								721215				
								721215				
								721215				
								721215				
								721215				
								721215				
								721215				
								721215		1		

190

TOPIC H.1 - IMMEDIATE COMMANDS

A. DATA SET NAME:

NASIS Message File

B. CREATED BY:

Not Applicable

C. TYPE OF FILE:

VISAM

D. ORGANIZATION:

Sequential

E. KEY IDENTIFIER (CONTROL FIELD):

The fifteen byte key is composed of the eight byte message key concatenated to the seven byte line number.

F. RECORD LENGTH:

V(132)

G. BLOCKING FACTOR:

Not Applicable

H. PURPOSE:

This data set contains the NASIS system messages used by the various modules for prompting and diagnostic messages.

TOPIC H.2 - IMMEDIATE COMMANDS

A. DATA SET NAME:

Strategy Data Set

B. CREATED BY:

RTSSTRT

C. TYPE OF FILE:

VISAM

D. ORGANIZATION:

Regional Sequential

E. KEY IDENTIFIER (CONTROL FIELD):

Strategy Name (16 characters)

F. RECORD LENGTH:

328

G. BLOCKING FACTOR:

Not Applicable

H. PURPOSE:

To contain the stored strategies and formats created by and used by the various NASIS commands.

TOPIC H.3 - IMMEDIATE COMMANDS

A. DATA SET NAME:

Strategy Display Format

B. CREATED BY:

RDBSTRT

C. TYPE OF FILE:

Screen Display

D. ORGANIZATION:

Header = STRATEGY name (centered)

Data Lines = full width, word split

Overflow Lines = indented three characters

Page Overflow = full record

E. KEY IDENTIFIER (CONTROL FIELD):

Not Applicable

F. RECORD LENGTH:

Not Applicable

G. BLOCKING FACTOR:

Not Applicable

H. PURPOSE:

To display the contents of the data lines comprising a stored strategy.

C3

TOPIC H.4 - IMMEDIATE COMMANDS

A. DATA SET NAME:

Strategy Names Display Format

B. CREATED BY:

RDBSTRT

C. TYPE OF FILE:

Screen Display

D. ORGANIZATION:

Data Lines = complete 16 character strategy names separated by two blanks (as many as will fit on a line).

E. KEY IDENTIFIER (CONTROL FIELD):

Not Applicable

F. RECORD LENGTH:

Not Applicable

G. BLOCKING FACTOR:

Not Applicable

H. PURPOSE:

To display the names of the strategies present in the strategy data set.

TOPIC H.5 - IMMEDIATE COMMANDS

A. DATA SET NAME:

User Profile Table

B. CREATED BY:

TS2-Supervisor

C. TYPE OF FILE:

Segmented Array

D. ORGANIZATION:

Segment - 1 (Synonyms) - sequential

Segment - 2 (Default Keywords) - sequential

Segment - 3 (Default-Data) - random

E. KEY IDENTIFIER (CONTROL FIELD):

Not Applicable

F. RECORD LENGTH:

V(32,768)

G. BLOCKING FACTOR:

Not Applicable

H. PURPOSE:

To contain the user defined synonyms and defaults.

TOPIC H.6 - IMMEDIATE COMMANDS

A. DATA SET NAME:

NASIS User Profile Dataset

B. CREATED BY:

TS2-Supervisor

C. TYPE OF FILE:

VSAM

D. ORGANIZATION:

Sequential

E. KEY IDENTIFIER (CONTROL FIELD):

Not Applicable

F. RECORD LENGTH:

V(32,768)

G. BLOCKING FACTOR:

Not Applicable

H. PURPOSE:

To contain the lists of user defined synonyms and defaults for a particular NASISID.

TOPIC H.7 - DATA RETRIEVAL

A. DATA SET NAME:

VERETAB Table.

B. CREATED BY:

The NASIS modules which prompt for commands.

C. TYPE OF FILE:

Table

D. ORGANIZATION:

PL/I Data Structure

E. KEY IDENTIFIER (CONTROL FIELD):

Not Applicable

F. RECORD LENGTH:

Not Applicable

G. BLOCKING FACTOR:

Not Applicable

H. PURPOSE:

1. VERETAB EXTERNAL CONTROLLED.
This table contains the information necessary to associate a set of valid verbs (commands) and their respective entry points.

2. #_ENTRIES BINARY FIXED.
This field contains the count of the number of valid entries in the list below.

2. SIZE BINARY FIXED.
This field contains the count of the number of entries that can be contained in the list below.

2. SYMBOLIC_ID CHARACTER (8).
This field contains the default symbol that can be used to define user written extentions to this list of verbs.

2. COMMAND (VERB_COUNT).
   This list is used to describe the commands recognized by the defining module.

   3. NAME CHARACTER (8).
      This field contains the command name.

   3. ROUTINE CHARACTER (8).
      This field contains the name of the entry point to be called when this command is entered.

1. VERB_COUNT BINARY FIXED.
   This field must be set to the maximum number of entries allowable in the verb list, before the table is allocated.

I. PL/I DECLARATION:

```
/*          GENERALIZED NASIS SYSTEM VERB TABLE                 */

DCL 1 VERBTAB EXT CONTROLLED, /*DEFINE THE VERB TABLE          */
      2 #_ENTRIES BIN FIXED,  /*DEFINE THE CURRENT SIZE       */
      2 SIZE BIN FIXED,       /*DEFINE THE MAXIMUM SIZE       */
      2 SYMBOLIC_ID CHAR(8),  /*DEFINE THE DEFAULT TERM       */
      2 COMMAND(VERB_COUNT),  /*DEFINE THE VERB ENTRIES       */
        3 NAME CHAR(8),       /*DEFINE THE VERB NAME          */
        3 ROUTINE CHAR(8);    /*DEFINE THE ROUTINE NAME       */

DCL VERB_COUNT BIN FIXED:     /*DEFINE THE TABLE SIZE         */
```